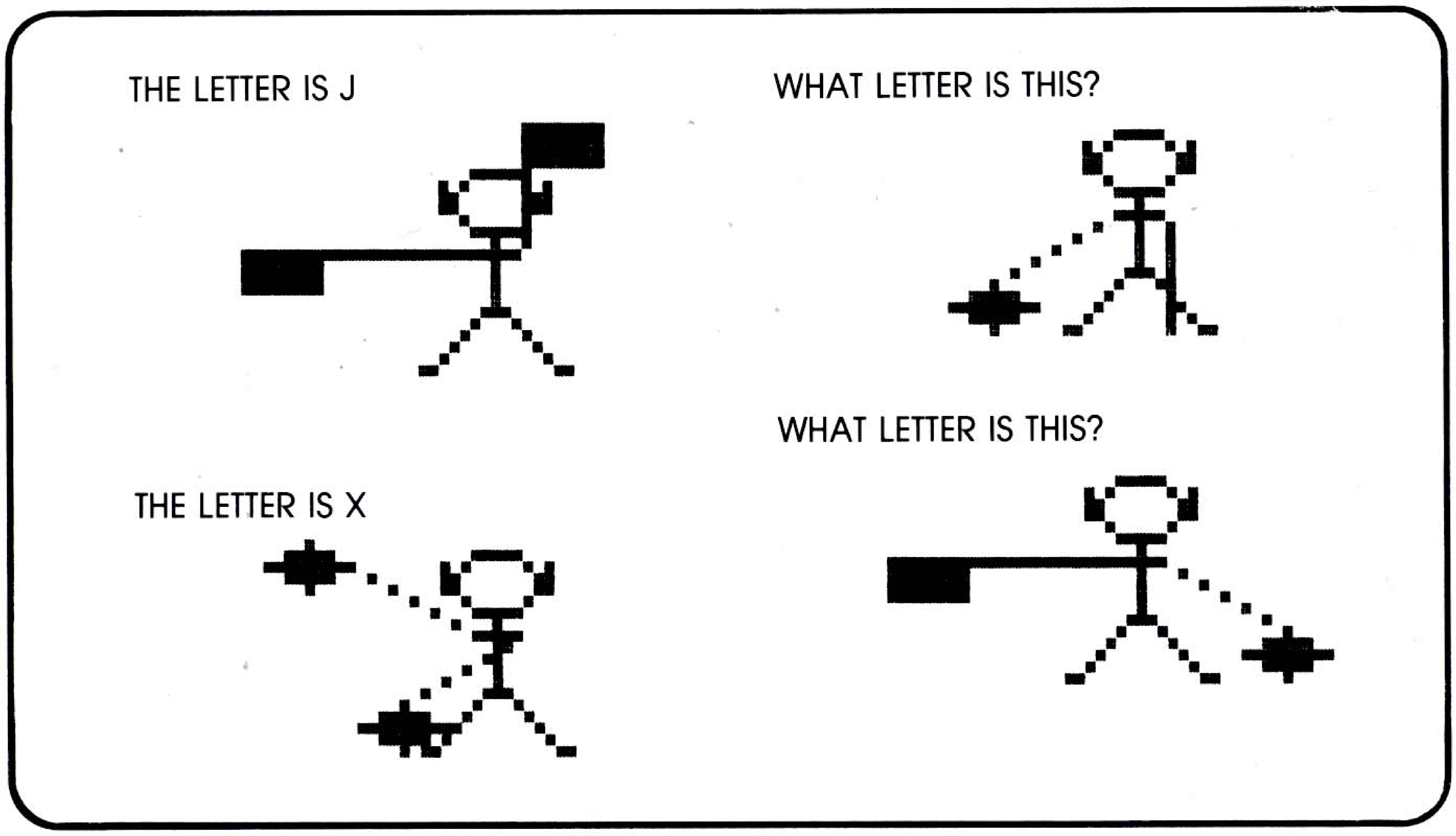
TRS-80 SYSTEM 80 VIDEO GENIE PMC-80

Issue 22, September 1981



TEACH YOURSELF SEMAPHORE

an invaluable aid to modern communications!

MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034. AUSTRALIA. TELEPHONE (08) 211 7244. PRICE: AUS. $2.50, N.Z. $4.00, U.K. £1.50

***** ABOUT MICRO-80 *****

EDITOR:	IAN VAGG
SOFTWARE EDITOR:	CHARLIE BARTLETT
HARDWARE EDITOR:	EDWIN PAAY
U.K. CORRESPONDENT:	TONY EDWARDS

MICRO-80 is an international magazine devoted entirely to the Tandy TRS-80 microcomputer and the Dick Smith System 80/Video Genie. It is available at the following prices (all prices shown in Aus.$ except for U.K. prices which are in pounds Sterling).

12 months subscription	Aus.	$24.00
	NZ.	$36.00 (Airmail)
	Hong Kong	$46.00 (Airmail)
	U.K.	£16.00
Single Copy	Aus.	$2.50
	N.Z.	$3.50 (Airmail)
	Hong Kong	$4.25 (Airmail)
	U.K.	£1.50
Months programs on cassette	Aus.	$3.50
	N.Z.	$4.00 (Airmail)
	Hong Kong	$4.50 (Airmail)
(at present available from Australia only)	U.K.	$4.75 (Airmail)
12 months subscription to magazine and cassette	Aus.	$60.00
	N.Z.	$78.00 (Airmail)
	Hong Kong	$88.00 (Airmail)
	U.K.	£41.00 (Airmail)

Special bulk purchase rates are also available to computer shops etc. Please use the form in this issue to order your copy or subscription.

The purpose of MICRO-80 is to publish software and other information to help you get the most from your TRS-80, System 80 or Video Genie and their peripherals. MICRO-80 is in no way connected with either the Tandy or Dick Smith organisations.

** WE WILL PAY YOU TO PUBLISH YOUR PROGRAMS **
Most of the information we publish is provided by our readers, to whom we pay royalties. An application form containing full details of how you can use your TRS-80 or System 80 to earn some extra income is included in every issue.

** CONTENT **
Each month we publish at least one applications program in Level I BASIC, one in Level II BASIC and one in DISK BASIC (or disk compatible Level II). We also publish Utility programs in Level II BASIC and Machine Language. At least every second issue has an article on hardware modifications or a constructional article for a useful peripheral. In addition, we run articles on programming techniques both in Assembly Language and BASIC and we print letters to the Editor and new product reviews.

***** CONTENTS *****

MICRO-80 is registered by Australia Post - Publication SQB 2207 Category B

AUSTRALIAN OFFICE AND EDITOR:

MICRO-80, P.O. BOX 213, GOODWOOD, SOUTH AUSTRALIA, 5034. TEL.(08) 211 7244

U.K. SUBSCRIPTION DEPT:

24 WOODHILL PARK, PEMBURY, TUNBRIDGE WELLS, KENT TN2 4NW.

Printed by:

Shovel & Bull Printers, 312A Unley Road, HYDE PARK, S.A. 5061

Published in Australia by MICRO-80, 433 Morphett Street, ADELAIDE.

***** EDITORIAL *****

Stories coming out of the U.S.A. suggest that the Model I TRS-80 is far from dead there, despite being withdrawn from the market on 31 December last because of excessive radio frequency interference. These machines are still manufactured and sold in Mexico and, reportedly, many thousands are finding their way across the border into the U.S.. It is also reported that the second hand value of a Model I exceeds it's new value and that anyone who advertises one is likely to besieged by hopeful buyers. In this environment, one would expect the PMC-80 (System 80/Video Genie) to be faring particularly well. However, as a result of a court action taken several months ago by Tandy, we understand that PMC-80's have been temporarily withdrawn from the U.S. market.

In Europe and Australia this is not the case, of course, and this computer is proving to be very popular. The Dick Smith organisation is taking advantage of the huge range of TRS-80 add ons which is available. The colour upgrade kit which was mentioned by our U.K. correspondent last month is rumoured to be under evaluation, brought back to this country by no lesser a person that Dick Smith himself when he returned from a recent trip to the U.K.. As it stands, this kit is thought to be rather too difficult to fit to have wide appeal and the U.S. ERAM high resolution graphics module is also under consideration. Dick Smith Electronics already sells the Percom Doubler and no doubt is assessing other peripherals and modifications.

For several months past, the glossy American computer mags have carried elaborate advertisements for a program produced in the U.K., called "The Last One". It is proudly proclaimed to be the last computer program you will need to buy since, with it you will be able to write virtually any other program just by answering a few plain language questions (the suggested price of US$600 would certainly make it rather unlikely that you could afford to buy any more programs for a while!). But, where was the program. No-one over here seemed to have seen it and foreign dealers did not seem to have it. Was it all an expensive spoof? We can now say emphatically, no. The program exists, at least for some computers and is very, very good. We recently saw it in action on a TRS-80 Model II. It was being used to develop an elaborate mail list program and cut the programmer's time by at least 70%. The BASIC code it produced was good, tightly packed, well commented and had excellent error handling routines built in. We understand that a version for the TRS-80 Model III is 2 - 3 months away but that there are no plans to develop a Model I version, which is a pity. You must have a disk system to use "The Last One". We look forward to obtaining one ourselves for our Model III as soon as they are available and will report more fully then.

One of the more embarrassing aspects of Micro-80 has been the way in which our publication dates have fallen behind over the past twelve months or so. We reached the point where we were a full three months behind. Regular readers will have noticed we are catching up and we will be producing a magazine every three weeks until we are once again on target, early in 1982. At which point we will all be much happier. The accelerated rate of progress has caught one or two of our authors a little unawares. Such is the case with the current series of constructional articles - Joysticks and Input/Output Ports for your '80. Part 3 of this series will appear in the next issue but you will not have too long to wait. One of the other casualties has been the Input/Output colomn which has been sparse of late. Many of our readers are receiving answers by mail but, nevertheless this is an important aspect of our magazine. So, Input/Output will once again become a significant and regular feature next issue and thereafter. You will have noticed that the pages given over to '80 Users Groups and Reader Requests have not appeared in the last two issues. This is because the vast majority of our readers have subscriptions and it does seem a waste of valuable space to repeat the same (almost) information every month. Therefore, both features will appear every second or third issue (turn and turnabout).

** DISK VERSION OF MICRO-80 **

Starting with this issue, the programs which appear in Micro-80 each month will be available on disk. Our index on page 36 now carries the Filespec of each program. The disks are recorded in 35 track single density format with their own operating system so will function in a single drive system. The cost of a twelve month magazine subscription plus disk is A$120, i.e. you will pay only $8 for a high quality disk complete with programs mailed to your address. Single issue disks may be purchased for $10 each. If you already have a magazine or cassette subscription, you may upgrade to a disk subscription by sending $8 (magazine subscribers) or $5 (cassette subscribers) for each month of your subscription remaining. For example, if you have already received five magazines from your twelve month subscription - send 7 x $8 = $56 to upgrade to the disk subscription for the remaining seven months.

** ORCHESTRA 80 **

We have been so impressed by the quality of music generated by the software and hardware which comprise the Orchestra 80 package that we have recorded a sample tune on the end of the cassette this month so that cassette subscribers can hear for themselves. The music you will hear was not produced by an electronic organ, it was produced by a TRS-80 Model I equipped with Orchestra 80. If you are not a cassette subscriber but would like to hear the music, send $2 for a cassette of sample tunes created by Orchestra 80.

- 0000000000 -

***** BETTER BASIC PROGRAMMING - PART 5 by Rod Stevenson *****

** ASSEMBLY **

PRELIMINARIES.

Perhaps a better title would be Assembly-Basic, for this is what I do with my own assembly routines - use them from a BASIC program. Of course, I do initially write and debug them in assembly, using the Microsoft Editor Assembler Plus. When they are running to my satisfaction I convert them to BASIC and POKE statements for loading from the "host" BASIC program. The actual mechanics of this is that, having assembled into memory using the "IM" command of EDTASM+ which retains the source-code for debugging (yes, even Eddy's don't always run first time), I return to BASIC and PEEK the locations where the program resides so I can write a DATA line with these numbers for the BASIC which is added later, i.e. the PEEKed values are on the screen while typing in the DATA line. Yes, I know Eddy has a routine to do this process automatically, but I am not so sophisticated! (Eddy is Edwin Paay - one day we will prevail upon him to publish this routine for us - Ed.)

While on the subject of lack of sophistication, I readily and openly admit that my own level of assembly capability is an immeasurable leap below that of Eddy Paay. But I like to excuse this as an advantage to you, our reader. Because Eddy fluently speaks Assembly, he has absolutely no idea that anyone else in the world should have the slightest difficulty. But we do. In fact, this article follows the format set by our elementary assembly group in the Adelaide Users' Group. And perhaps here is the place to offer a suggestion to other groups - start your own elementary sessions. There's no need to have a top-notch experienced Assemblerist to conduct it: I run the Adelaide Group's! Indeed, we all learn from each other, and find considerable enjoyment doing so. Actually the whole thing about Assembly is to think in the right way - the actual detailed facts (instructions, etc.) will come easily then. The same applies to BASIC and the whole area of computing. Should any other group be interested, I will be pleased to provide further details - for extended ramblings, send a cassette and you'll be more likely to get full details than if I have to write it all.

INTRODUCTION.

As foreshadowed in the earlier episodes, this is not intended to be a complete tutorial on Assembly. I expect that, by now, you will have taken my advice and obtained Bill Barden's "TRS-80 Assembly Language Programming". I also expect that you will have read it and, hopefully, have understood at least some of it! If I am expecting too much already, I suggest you rush out to your local Tandy store and get a copy. That which follows assumes that you are familiar with at least the first three chapters.

The other book suggested is "Z80 Instruction Handbook" from Scelbi Publications. I use this book all the time as a reference, both to find out if an appropriate instruction exists and to get its exact syntax and full explanation for complete understanding of any side-effects. If you can't get this, a (poorer) next choice would be "Zilog Z80 Programming Reference Card".
"Z80 Instruction Handbook" is available from MICRO-80 - subject to supply.
ELEMENTARIES.

Although some knowledge of assembly language is assumed, I will set down briefly a few pertinent points for you to further investigate and think on until you've really got a "feel" for the concepts. With Assembly, as with all other computing subjects, it is definitely the way of thinking that matters, not so much the facts! The facts you can easily get from a reference book, but if you don't understand what the whole thing is about, you won't achieve much from the reference book. So stick with it and a true understanding of Assembly will come. When it does, you too will get great satisfaction from writing in Assembly.

Perhaps now is a good time to define the difference between Assembly and machine-code. When people say they write in machine-code, they really mean Assembly, and indeed the two are used interchangeably, although really they are not the same thing. Machine-code is the binary digits the Z80 understands; Assembly is the mnemonics and op-codes the ssembler accepts and then Assembles into machine-code. For while 11ØØ1ØØ1 is the machine-code for the Assembly mnemonic RET, which is easier to remember? Even if you realise that 11ØØ1ØØ1 is C9 in hexadecimal, it's still easier to remember and write in Assembly using RET. Even if you don't use an Assembler, but look up tables to convert your mnemonics to machine-code (hand-assemble), are you not still writing in Assembly?

POKE and PEEK are the Basic language commands for accessing memory directly. POKE simply loads a value into memory, PEEK gets it out of memory. So to think there is something complicated or that POKE and PEEK belong to some separate language, you can see is totally erroneous.

If you have followed my advice and read Barden's book, you will have come upon the seemingly difficult subject of flags. Of course, they are not difficult at all in reality, and are essential to Assembly programming. So, if you didn't follow them in all their magnificence, try again! They are so important though, that I will present here yet another way of explaining them - my own!

Just as BASIC has a facility for comparison (IFA=B THEN...), so Assembly has flags to allow you to write code dependant on the result of arithmetic operations. By the way, initially you'll get into less trouble if you use the A register for all arithmetic operations. The flags which you must know and understand are the zero and carry flags. There are others, but you'll get by (at least initially) with just these two, which are in reality four, since either the zero flag is set or not, and either the carry flag is set or not. In reality, what you say is "if the zero flag is set then...". The zero flag will be set if the result of the arithmetic operation you just performed on the accumulator was zero; it will be reset if the result of the operation you just performed on the accumulator was not zero, i.e. Z or NZ. Similarly, with the carry flag - C or NC. Although there is a slight complication here as the carry flag also acts as a borrow flag for subtraction operations on the accumulator.

The reason why Carry and Zero flags are the ones to use initially is that they are the only ones to use with relative jumps - viz. JRNC,LABEL. It is advisable to use relative jumps wherever possible to maintain relocatability of your program.

THE ASSEMBLER.

The Assembler I use is the Microsoft Editor Assembler Plus (available from MICRO-80 PRODUCTS). It has various useful additions to the Tandy version, but it also has a monitor (ZBUG)and a really terrific manual included with it. And you really do need a manual - Tandy's version doesn't have one which tells you how to work the Assembler, it only has pages and pages of Z80 op-codes, the explanation of which is almost impossible to understand. I firmly believe this is why so many are deterred from Assembly language programming.

SYSTEM 80s.

I have found that some System 80s (two to date) have a problem recording the assembled object code on tape and loading it back as a SYSTEM program. I don't know why, and can't advise you what to do. Ask Dick!

```
                 00100 ;SAMPLE SIMPLE ASSEMBLY PROGRAM WRITTEN
                 00110 ;BY ROD STEVENSON TO WHITE-OUT SCREEN.
                 00120 ;
                 00130 ;CAN BE CALLED FROM BASIC BY USR(0).
                 00140 ;WILL RETURN TO CALLING PROGRAM AFTER RUNNING.
                 00150 ;
7F00             00160         ORG     32512
7F00 21003C      00170 INIT    LD      HL,3C00H        ;SCREEN START ADDRESS
7F03 0E10        00180         LD      C,16            ;VERTICAL LINES
7F05 0640        00190 VRTLP   LD      B,64            ;HORIZONTAL SPACES
7F07 36BF        00200 HRZLP   LD      (HL),191        ;GRAPHICS BLOCK
7F09 23          00210         INC     HL              ;NEXT SCREEN SPACE
7F0A 10FB        00220         DJNZ    HRZLP           ;64 SPACES ACROSS
7F0C 0D          00230         DEC     C               ;NEXT LINE
7F0D 20F6        00240         JR      NZ,VRTLP        ;16 LINES
7F0F C9          00250         RET                     ;BACK TO BASIC
7F00             00260         END     INIT
00000 TOTAL ERRORS
```

This program is not intended as a masterpiece. In fact, Eddy Paay, in his series on machine language, had a much more efficient and neater one using the LDIR instruction. The reason I wrote this one in this way, is that I imagine readers of this article are just into Assembly from BASIC, and this particular method is very similar to the approach which would be used in BASIC.

The whole reason for doing this in Assembly is speed; remember, the two major and common reasons for Assembly are speed and changes to the operating system which must be done from Assembly. The latter is the subject of the next example. Incidentally, PRINTSTRING$ will perform the same almost as quickly from BASIC.

Now to an explanation:

Line 16Ø	is an Assembler pseudo-op that tells the assembler where to put the code. See the Editor Assembler Plus manual for details.
Line 17Ø	is the actual start of the program, and I always give the first line a label - you'll see why later. This line loads the first screen address into HL.
Line 18Ø	loads C as a counter for the 16 lines on the screen.
Line 19Ø	also has a label and loads B as a counter for the 64 spaces across the screen. Notice you can use decimal and hex notations quite happily together and the Assembler looks after all!
Line 2ØØ	with a label loads 191 (the value of the solid graphics block) into the memory location (which happens to be the screen) pointed at by HL .
Line 21Ø	gets HL to point to the next screen position.
Line 22Ø	is the only non-BASIC type instruction. It decrements B and if B does not become zero will send the program back to the address specified - in this case, a label, so that loop HRZLP (for horizontal loop - remember my urging to use meaningful names) will continue for 64 spaces, then B will become zero and the program will fall through to line 23Ø.
Line 23Ø	decrements C to point at the next line down the screen.
Line 24Ø	will test if C has become zero (by the zero flag), and if not, will send the program to the address specified - in this case, the label VRTLP (for vertical loop) until 16 lines have been done, when the program will fall through to line 25Ø.

Line 25Ø simply returns the program to BASIC (or whatever program called this routine). If you want to run this on its own, just give this line a label (such as LOOP), then change RET to JR, and in the next column put LOOP. Then it will run and stay there until you press RESET! In the op-code this is 18,FE or 24,254 decimal in BASIC.

Line 26Ø is another Assembler pseudo-op (if you don't put it in you'll get an error) which tells the Assembler you have finished. The label INIT (see why I put one in the first line when it wasn't used in the program) is to make the Assembler record on tape the entry point (in this case, the start) of the program, so when you type / (slash) after loading the system tape you don't have to put an address.

So, I hope it's all clear so far. (No, not a pun). Can you see the BASIC two, nested for-next loops? 2ØØ to 22Ø is inside 19Ø to 24Ø.

Now, onto the real subject of this article - using it in BASIC.

```
10 'BASIC PROGRAM BY ROD STEVENSON TO USE WHITE-OUT PROGRAM WRITTEN IN ASSEMBLY.
20 TM=PEEK(16561)+PEEK(16562)*256    'GET TOP OF MEMORY
30 AD=TM-20    'TOP MEM, MINUS SIZE PROGRAM, MINUS 2
40 POKE16561,AD-INT(AD/256)*256:POKE16562,INT(AD/256):CLEAR50    'PROTECT NEW
    MEMORY SIZE
50 AD=PEEK(16561)+PEEK(16562)*256+2    'NEW MEMORY SIZE
60 EA=AD    'SAVE ADDRESS IN CASE CHANGED BY NEXT LINE
70 IFAD>32767THENAD=AD-65536    'SYNTAX REQUIRED IF >16K OF MEMORY
80 FORI=ADTOAD+15:READD:POKEI,D:NEXT    'POKE M/L INTO MEMORY
90 POKE16526,EA-INT(EA/256)*256:POKE16527,INT(EA/256)    'POKE M/L ADDRESS INTO
      USR(0)
100 DATA33,0,60,14,16,6,64,54,191,35,16,251,13,32,246,201    :'M/L PROGRAM
110 X=USR(0)    'CALL M/L PROGRAM
120 IFINKEY$=""GOTO120    'AFTER M/L WAIT FOR KEY-PRESS
130 CLS
140 IFINKEY$=""GOTO140    'AFTER CLS WAIT FOR KEY-PRESS
150 GOTO110
```

Alright, so this looks complicated! It's not! The only "frill" is that I've let the computer determine how much memory it's got and protect enough for this program. After that, I simply POKE it in and access it. The reason for doing so is that your masterpiece is not dependant on the operator remembering to protect memory at the right place.

There are other ways of storing and accessing machine-code routines, but they'll have to wait until next time.

So, now an analysis:

Line 2Ø finds out what the top of memory is currently set to. Note, TM as a meaningful variable name. Of course, this top of memory will vary, not only with the amount of memory, but with whatever memory is protected for other programs in answer to the Memory Size? question.

Line 3Ø allows room for the program plus extra. When you specify a memory size it actually protects more than specified, because the top of memory is actually the highest memory location BASIC can use; but when using this method of protecting memory, you'll get exactly what you ask for.

Line 4Ø sets the new memory size. A full explanation of the 2 byte memory addressing used by the Z80 will be offered next instalment, but briefly: the memory address must be held in 2 bytes (it's too big to fit into one), and so is divided into most and least significant parts, the former being 256 times bigger than the latter. So the whole address divided by 256 is the larger number, the remainder is the smaller. Having set the new memory size (protection), it is necessary to CLEAR a number of bytes to reset pointers. And more on pointers next time. Yes, it is the same CLEAR as in clearing string space, and the same CLEAR which zeroes all variables.

Line 5Ø is necessary because the CLEAR in line 4Ø destroyed the variable which held the memory protected for our program. So we have to find it again by the same process as used in line 2Ø.

Lines 6Ø and 7Ø take care of the syntax required for more than 16K of memory, so don't worry too much about these two lines here.

Line 8Ø reads the machine-code in DATA statement line 1ØØ and POKEs it into memory where it has been protected already. A trap for young players here is to put too big a loop in the for-next counter. Realise that the very first location will have a value POKEd into it! After all, Ø-15 is actually 16 places!

Line 9Ø tells USR(Ø) in line 11Ø where to find the routine.

Line 11Ø calls the program by jumping to the address given it by line 9Ø. So there can be more than one machine language program in memory at once and whatever address is given in 16526-7 will determine which one will be jumped to by USR(Ø).

Line 12Ø is where the program will return after executing the machine-code. It will loop here till a key is pressed when it will clear the screen in line 13Ø, wait for another key press in line 14Ø, then call the routine again by jumping to line 11Ø.

Obviously this program on its own is not very exciting; it's meant only to illustrate various points - one being the simplicity of writing Assembly and using it from BASIC!

Incidentally, those who wonder what's happened to the stack (more on this next time for those who haven't wondered) when I reset Memory Size? The BASIC interpreter handles a line at a time; so as long as the new memory protection is set in a stand-alone line, all is well.

```
                 00100 ;
                 00110 ;ROUTINE WRITTEN BY ROD STEVENSON TO PROVIDE
                 00120 ;LINE INPUT FOR LEVEL 2.
                 00130 ;
                 00140 ;TO BE CALLED FROM BASIC WITH THE ADDRESS OF THE INPUT
                 00150 ;STRING WHICH HAS BEEN ALLOCATED PASSED AS THE
                 00160 ;ARGUMENT IN USR(0).
                 00170 ;LENGTH OF THE INPUT WILL BE RETURNED IN X OF X=USR(0).
                 00180 ;
7FBC             00190         ORG     32700
7FBC D9      •   00200 INIT    EXX             ;EXCHANGE REGISTERS
7FBD CD7F0A      00210         CALL    0A7FH   ;GET ADR FROM USR(0)
7FC0 0E00        00220         LD      C,0     ;ZERO C FOR COUNTER
7FC2 CD4900      00230 GETCHR  CALL    0049H   ;GET CHAR FROM KEYBD
7FC5 FE0D        00240         CP      13      ;CHECK IF ENTER
7FC7 280D        00250         JR      Z,RETN  ;IF ENTER RETURN TO BASIC ROUTINE
7FC9 CD3300      00260         CALL    0033H   ;PRINT CHAR ON SCREEN
7FCC 77          00270         LD      (HL),A  ;PUT INTO $TRING MEMORY
7FCD 0C          00280         INC     C       ;ADD TO COUNTER
7FCE 79          00290         LD      A,C     ;CHECK NO. INPUTS
7FCF FEC8        00300         CP      200     ;ONLY 200 ALLOWED
7FD1 3003        00310         JR      NC,RETN ;IF >200 RETN TO BASIC
7FD3 23          00320         INC     HL      ;INCREMENT $TRING LOCATION
7FD4 18EC        00330         JR      GETCHR  ;GET ANOTHER CHAR
7FD6 79          00340 RETN    LD      A,C     ;RTN TO BASIC ROUTINE:COUNT TO A
7FD7 D9          00350         EXX             ;GET BACK REGISTERS
7FD8 2600        00360         LD      H,0     ;ZERO H
7FDA 6F          00370         LD      L,A     ;COUNTER TO H
7FDB C39A0A      00380         JP      0A9AH   ;PUT COUNTER INTO X OF USR(0) &
                               ;RETURN TO BASIC PROGRAM
7FBC             00390         END     INIT
00000 TOTAL ERRORS
```

This is a source-code for the line-input program provided in the input checking episode. Again, only as an example, this one is in Assembly because it can't be done in BASIC - although something like it (but not as good!) can be built up by adding together INKEY$. The routine is not written in a BASIC-style, so here's your chance to break the BASIC-thinking habit!

By now, you will have become familiar with my way of writing and thinking (horrors!), so I wont' analyse every line. The comments should tell enough, so just a few specific points:

Line 2ØØ exchanges registers in case this routine is called from the middle of a BASIC line, in which case certain registers will be holding information required by the interpreter for the remainder of that line. They are changed back again by line 35Ø, after a bit of manipulation in line 34Ø to save the counter value in C. The counter in C is, in this case, counting upwards as each character is input - in the previous screen writing program the counters were counting downwards. We'll leave our readers to ponder the significance of this difference and the ramifications thereof, and decide which is better - it could well be either way in either program.

Lines 23Ø and 26Ø are ROM routines found in Eddy Paay's "ROM Reference Manual" (from MICRO-80 PRODUCTS), and frankly, without it I couldn't have written this program!

Lines 21Ø and 38Ø are ROM calls associated with the USR(Ø) which are documented (almost well enough) in the Level II manual.

```
10 'BASIC ROUTINE WRITTEN BY ROD STEVENSON TO USE "LINE-INPUT" PROGRAM WRITTEN
      IN ASSEMBLY
20 'ROUTINE TO ACCEPT ANY INPUT (MAXIMUM 200 CHARACTERS) UNTIL TERMINATED BY
      ENTER
30 '
40 POKE 16561, 186 : POKE 16562, 127   'PROTECT MEMORY
50 CLEAR 500   'RESET POINTERS & ALLOW ENOUGH $TRING SPACE
```

```
60 CLS
70 FOR I=32700 TO 32733
80 READ ML
90 POKE I, ML    'PUT M/L INTO MEMORY
100 NEXT
110 DATA 217, 205, 127, 10, 14, 0, 205, 73, 0, 254, 13, 40, 13, 205, 51, 0, 119,
          12, 121, 254, 200, 48, 3, 35, 24, 236, 121, 217, 38, 0, 111, 195, 154,
 10
120 POKE 16526, 188 : POKE16527, 127    'PUT M/L ADDRESS INTO USR(0)
130 IN$=STRING$(200, 65)    'CREATE $TRING TO HOLD INPUT
140 AD=VARPTR(IN$)    'GET ADDRESS OF $TRING ADDRESS
150 AD=PEEK(AD+1)+PEEK(AD+2)*256    'CONVERT $TRING ADDRESS TO A DECIMAL NUMBER
160 PRINT"INPUT UP TO 200 CHARACTERS?"
170 X=USR(AD)    'PASS ADDRESS OF $TRING TO M/L ROUTINE
180 PRINT
190 PRINT"THE INPUT WAS: "
200 PRINT LEFT$(IN$, X)    'PRINT ONLY NUMBER OF CHARACTERS INPUT
210 PRINT
220 GOTO 120
```

And this is the BASIC program to use the machine-language routine. Again, not a great deal of analysis required, I hope. In this case, memory is protected from BASIC, but 16K is assumed, i.e. top of memory is not looked for as in the last BASIC program to white out the screen. To change the location of this routine, it's simply a matter of changing the loop numbers in line 7Ø, and the address given to USR(Ø) in line 12Ø, and protecting memory in line 4Ø or manually, by answering Memory Size? Actually, this BASIC program is only the same one I presented two episodes ago in the input-checking routine. The "bones" of that program which were in lines 1 and 3276Ø have been UNPACKed with PACKER for easier reading and understanding (I hope).

CONCLUSION.

Almost the end of this article. I apologise for getting carried away in my enthusiasm for Assembly. As always, feedback from you, our readers is welcome. We are nearing the end of the scheduled series and if it is to continue past the scheduled subjects, we need to know what is required. Please feel free to make comments and requests for further information.

Next month we will cover the following topics: Explanation of the stack, ways of storing machine-code, handling and disabling break-key (promised in first episode and not forgotten), calls other than USR(Ø), pointers, fixed RAM, VARPTR, monitors and utilities, two byte addressing conventions and decoding. Sound routines and explanation will be the subject of another, simple, episode.

- 0000000000 -

***** THE THEORY AND TECHNIQUES OF SORTING - Part 1 by B. Simson *****

This is the first article of a series discussing the techniques of and the theory behind sorting routines. This first article will be oriented towards the beginner. I will assume that you do not know anything about sorting at all, and your level of knowledge of BASIC is fairly elementary.

Let's say that you were given the task of sorting some numbers, starting with one number, say 35. It's easy. The answer is: 35. No sorting needed at all! Let's assume that you now had to write a program on your computer to sort 2 numbers. First, you should give some thought as to how you will store them in a computer. The first way that comes to mind is by using some variables, say A and B. So, one way to go about it is this:

```
10 INPUT A,B
20 IF A<B THEN PRINT A,B ELSE PRINT B,A
```

Fine! Works well, is short and easy. But now, let's consider the problem of sorting 3 numbers.

```
10 INPUT A,B,C
20 IF A<=B THEN 80
30 REM    A IS > B HERE, SO NOW WE HAVE TO DETERMINE
40 REM    WHERE C GOES.
50 IF C<=B PRINT C,B,A : END
60 IF C<=A PRINT B,C,A : END
70        PRINT B,A,C : END
75
80 REM    A IS < OR = B, SO WHERE DOES C GO?
90  IF C<=A PRINT C,A,B : END
100 IF C<=B PRINT A,C,B : END
110         PRINT A,B,C : END
120 REM   PHEW !!
```

Another way to do this is to consider each permutation of the result individually. With 3 numbers, we have 6 permutations, that is...

A,B,C
A,C,B
B,A,C
B,C,A
C,A,B
C,B,A

So we could write something like this:

```
10 IF A<=B AND B<=C PRINT A,B,C : END
20 IF A<=C AND C<=B PRINT A,C,B : END
          :                 :
          :                 :
60 IF C<=B AND B<=A PRINT C,B,A : END
```

However, it should quickly become apparent that such a method is undesirable for larger list sizes. For example, a list of just 6 numbers has 720 permutations, and a list of just 10 numbers has 3,628,000 permutations! The above method would result in some very large programs for relatively small list sizes.

What we need is a method of problem solution using a predefined sequence of steps applicable to a general list size, that is ..... an algorithm.

Some other definitions are in order:

Sorting: The operation of arranging items into some sequential order according to some ordering criterion, e.g. the ordering criterion for the alphabet has been established to be A before B, B before C, etc.

Pass: A term used to describe a part of the operation of a sort. It is the operation whereby all items in a given list are examined individually in an orderly fashion.

One fairly elementary algorithm for sorting an unknown list size is called the "bubble" or "ripple" sort. We will now examine this algorithm in detail, using as our example to work from, a list size of 6 numbers (or items). Let's say our list is comprised of the following items:

8 4 3 9 6 5

We will need a structure for storing the data. This requires a structure of homogeneous data elements (i.e. of the same type) that facilitates easy access. Such a structure is available in BASIC, called an array.

So, let's store the first item in the first "cell", the second in the second "cell", etc. We shall call the whole array or family of items ...A, so that the first item (the number 8) is referenced by A(1), the second item by A(2), etc.

Now, looking at the list of numbers above, there are 5 adjacent pairs of items (assuming the list is not circular), that is... 8 & 4, 4 & 3, etc. This algorithm will examine each adjacent pair and swap them if necessary, i.e. if left item is greater in value than the right item. This is what is known as sorting by transposition.

So the sequence of steps will be:

1. Compare 1st pair, then 2nd pair, 3rd pair, 4th pair and finally 5th pair, i.e. examine A(1) and A(2) (values 8 & 4), then A(2) and A(3) (values 4 & 3), etc. If the result of any of the comparisons is that the left right, then the items in the pair shall be swapped. This whole step is one "pass". The last pair shall be called the "pair limit".

2. Now compare 1st pair, 2nd, 3rd and 4th, i.e. only the first 4 pairs are examined. The pair limit has been decremented by 1. Items shall also be swapped as in step 1.

3. Now compare only the first 3 pairs.

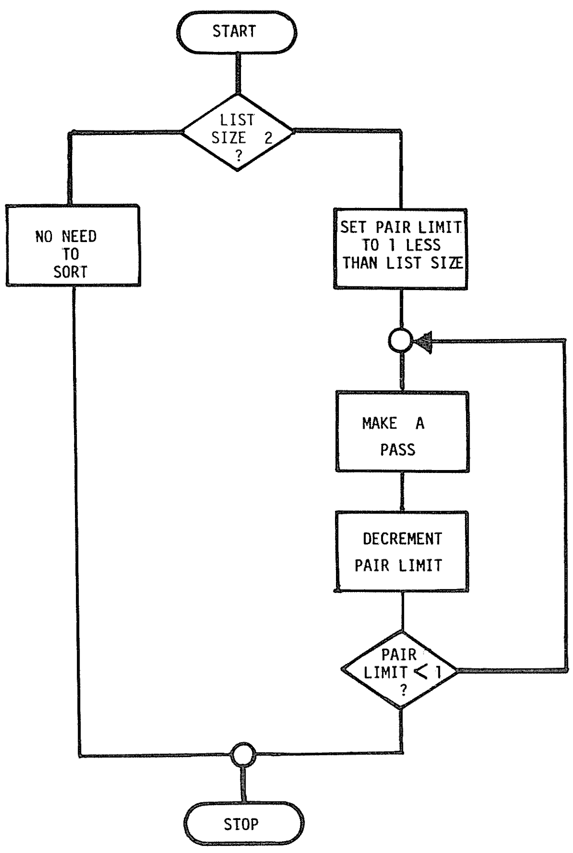
4. Now compare only the first 2 pairs.

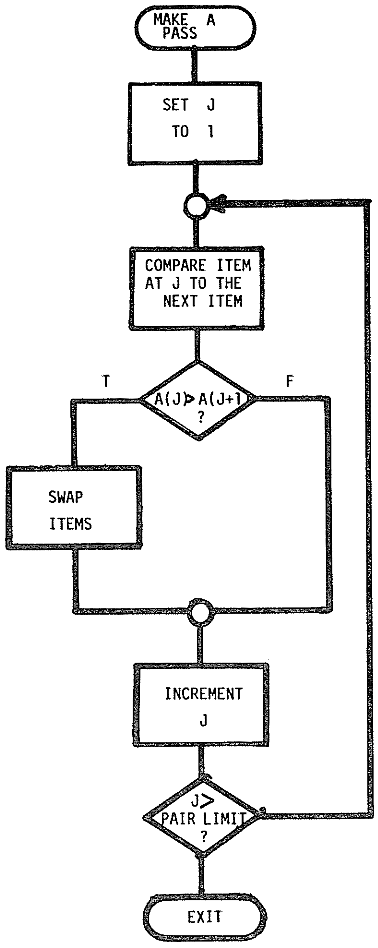
5. Lastly, compare just the first pair.

The above steps involved 5 passes, with the pair limit being reduced by 1 for each pass. The reason why the last pair does not need examination in step 2 is because the item having the largest value (9) will drop to cell 6 after the first pass, so only the first 4 pairs need be examined after that, and so on.

Let's trace the workings of this algorithm on the data for the first pass. The flowchart showing the steps involved in the bubble sort algorithm should help you to follow the logic of the first pass.

BUBBLE SORT ALGORITHM





ITEMS COMPARED	SWAP?	LIST AFTER COMPARING
1st pair: 8, 4	4, 8	4 8 3 9 6 5
2nd pair: 8, 3	3, 8	4 3 8 9 6 5
3rd pair: 8, 9	NO	4 3 8 9 6 5
4th pair: 9, 6	6, 9	4 3 8 6 9 5
5th pair: 9, 5	5, 9	4 3 8 6 5 9

If the list was arranged vertically, you can see that each successive pass causes the smaller valued items to 'bubble' up to the top, and the larger ones to sink to the bottom.

Now, let's write a basic program to do this. First, let's assign some variable names:

N = List size (6 in this case)
A = Name of array holding all items in list
J = Index into array A. (method that items are accessed).
Temp = Temporary variable needed to swap a pair of values.

```
100 FOR PAIRLIMIT=(N-1) TO 1 STEP -1
120      FOR J=1 TO PAIRLIMIT
140           IF A(J) > A(J+1) THEN
                      TEMP=A(J):A(J)=A(J+1):A(J+1)=TEMP
160      NEXT J
180 NEXT PAIRLIMIT
```

Line 1ØØ causes the pairlimit to range from 5 down to 1, as required above.
Line 12Ø enables the first 5 (pairlimit) pairs to be accessed, using J.
Line 14Ø compares an adjacent pair.

If out of sequence, then the items in the pair are swapped, using Temp to achieve this.

The outer FOR/NEXT loop is one pass. For each pass, you can see that the pairlimit is reduced by 1, so that the inner FOR/NEXT loop examines one less pair each time.

There are many refinements that could be made to this algorithm. One of them is to use a flag to indicate that the data is sorted, so that any further passes are superfluous. If, for instance, after performing a pass, we have not swapped any items, this indicates that all the items are in the desired sequence. We do this by setting a flag when a swap is made, and testing if the flag is set after a pass is complete.

Also, we should test to see if the list size is not less than 2. If so, no pairs can be compared, and therefore no sorting is required.

Let's add these to our program:

```
90 IF N<2 THEN PRINT "NO SORT REQUIRED":GOTO 200
100 FOR PAIRLIMIT=(N-1) TO 1 STEP -1
110      FLAG=0
120      FOR J=1 TO PAIRLIMIT
140           IF A(J) > A(J+1) THEN
                       TEMP=A(J):A(J)=A(J+1):A(J+1)=TEMP:
                       FLAG=1
160      NEXT J
170      IF FLAG=0 THEN PAIRLIMIT=1
180 NEXT PAIRLIMIT
200 RETURN
```

Line 11Ø resets the flag ready for the next pass.
Line 14Ø sets the flag if a swap is made.
Line 17Ø tests whether a swap was made. If not, then the sort is complete, so we make an orderly exit from the routine by forcing an end to the outer loop.
Line 2ØØ simply makes this a subroutine so that we can call it later using GOSUB.

Now, let's put this routine into practice. First, we will generate some random data to be sorted, then print them out for you to see before sorting. Then the sort routine will be called, after which the sorted data will be printed.

```
10 CLS:INPUT "NUMBER OF ITEMS TO SORT";N
20 IF N < 1 THEN 10
25 RANDOM: DIM A(N)
30 FOR I=1TON
35      A(I)=RND(100) :  REM FILL WITH A NUMBER 1-100
40 NEXT I
45 REM  PRINT DATA BEFORE SORT
50 FOR I=1TON
52      PRINT A(I);
54 NEXT I: PRINT
56 REM  NOW SORT THE ITEMS IN THE LIST
58 GOSUB 90
60 REM  NOW PRINT THE SORTED LIST
62 PRINT "LIST AFTER SORT:"
64 FOR I=1TON
66      PRINT A(I);
68 NEXT I
```

ENTER the entire program, including the sort subroutine, and run it.

Try working through a list size of 2 in your mind, to see how that compares with the logic of the program at the start of this article. Neat, isn't it?

Next month, we will be looking at algorithms that involve sorting by exchange and sorting by insertion, and at a technique that can be used to speed up the old bubble sort algorithm.

- 0000000000 -

***** SOFTWARE SECTION *****

***** ROVING TARGETS LI/4K by Br. P. Van Eeken *****

This program is designed to teach children equivalence in fractions by rewarding correct answers with the opportunity to play the game Roving Targets.

Up to 24 blocks are drawn on the screen, some shaded in, some not. The student must answer the fractions that are not shaded in and then convert this fraction to an equivalent fraction. Having obtained the correct answer, the student may then fire at the Roving Targets which are two of the boxes that move randomly around the screen. At the same time, the computer fires at the student's "man". The objective is to score more hits on the targets than are scored on the student's man.

Line 5ØØ Chooses a number from 3 to 24, i.e. the fraction for use. (B) This line also finds the highest equivalent fraction or factor of this number (besides the number itself). (c)

Line 51Ø A random factor, that changes C, (if rnd(2)=1) from the highest factor of B to a lower factor, producing a lower equivalent fraction. The actual change occurs in lines 512, 513.

For example: If B=18 line 5ØØ will choose the value of C as being 9. (Being the highest factor or equivalent fraction for B, besides B itself).

If line 51Ø decides to choose a lower factor then lines 512 and 513 will determine that number... 3 in this case.

Hence the original number B(18) will be equated with C(3) and the fraction equated by the computer with eighteenths will be sixths; as B/c equals 6.

Line 526 makes J equal to B/C; i.e. 6
and N determines the number of sixths that are to be shaded in. I equals J*N and is the maximum number of eighteenths that are not to be shaded in.

Lines 529 to 540 then draw the number of fraction units B, in lots of the equivalent fraction C (sixths) and shades in the number of eighteenths B-I.

```
1 C.:P.A.896;:G.499
2 D.W,HALVES,THIRDS,QUARTERS,FIFTHS,SIXTHS,SEVENTHS
3 D.EIGHTHS,NINTHS,TENTHS,ELEVENTHS,TWELFTHS,THIRTEENTHS
4 D.FOURTEENTHS,FIFTEENTHS,SIXTEENTHS,SEVENTEENTHS,EIGHTEENTHS
5 D.NINETEENTHS,TWENTIETHS,TWENTY FIRSTS,TWENTY SECONDTHS
6 D.TWENTY THIRDS,TWENTY FOURTHS
7 C.:V=0:U=0:Q=0:T=10:REST.:F.X=1T.24:REA.A$:N.X
8 P.A.24;"< ROVING TARGETS >
9 D.@,UP,ENTER,DOWN,],LEFT,^,RIGHT
10 F.X=1T.4:REA.A$,B$:P."PRESS ";A$;" TO SHOOT ";B$:N.X
11 W=1:A(W)=50:A(2)=18:GOS.970:P.
12 P."THIS IS YOUR MAN...":GOS.970:A(2)=24:GOS.930:P.
13 P."THIS IS MY MAN.....":GOS.930
14 P.:P."PLEASE GIVE ME A NUMBER BETWEEN 1 AND 999
15 I."THE SMALLER THE NUMBER THE HARDER THE GAME";S
16 IF(S<1)+(S>999)G.8
30 C.:W=1:GOS.980
31 A(W)=R.(121):A(W+4)=A(W):A(W+1)=R.(36)+3:A(W+4)=A(W+1)
32 IF(A(W)<A(W+2)+7)*(A(W)>A(W+2)-7)G.31
33 IFW=3GOS.970:F.Y=1T.S:N.Y:G.44
34 GOS.930
35 F.X=A(2)-1T.A(2)+1:IF(X=A(4))*(A(1)<A(3))R.(2,0):G.45
36 IF(X=A(4))*(A(1)>A(3))R.(0,0):G.45
37 N.X
40 IF(A(3)>A(1))*(A(4)>A(2))R.(6,0)
41 IF(A(3)<A(1))*(A(4)<A(2))R.(4,0)
42 G.45
44 ON(P.(2,0))G.50,45
45 IFP.(0,0)=0GOS.200
46 IFP.(6,0)=0GOS.300
47 IFP.(4,0)=0GOS.400
48 IFP.(2,0)=0GOS.100
50 W=W+2:IFW>4W=1
60 GOS.950:IFW=3GOS.980
70 G.31
```

```
100 Y=A(W+1)+1
110 F.X=A(W)+7T.127:IFP.(X,Y)G.800
120 S.(X,Y):R.(X,Y):N.X:G.810
200 F.X=A(W)-1T.3S.-1:IFP.(X,A(W+1)+1)G.800
205 S.(X,A(W+1)+1):R.(X,A(W+1)+1):N.X
250 G.810
300 Y=A(W)+3
310 F.X=A(W+1)+3T.41:IFP.(Y,X)G.800
320 S.(Y,X):R.(Y,X):N.X:G.810
400 Y=A(W)+3
410 F.X=A(W+1)-1T.1S.-1:IFP.(Y,X)G.800
420 S.(Y,X):R.(Y,X):N.X:G.810
499 F.Q=1T.3
500 A=0:B=R.(22)+2:F.P=B-1T.2S.-1:IFB/P=I.(B/P)C=P:G.510
501 N.P:G.500
510 IF(C<13)+(C>1)*R.(2)=1G.520
512 F.P=C-1T.2S.-1:IFC/P=I.(C/P)C=P:IfC>1G.520
513 N.P:IF(C<2)+(C>12)G.500
520 D=-8:E=0:A=R.(C-1)+1:F.I=CT.1S.-1:IFC/A=.(C/A)G.526
521 N.I:G.520
526 J=B/P:N=R.(J):I=J*N:A=0MF.P=1T.BS.C:F.O=1T.C:W=1:IFP>IW=3
530 D=D+8:A(W)=D:A(W+1)=E:IFW=1GOS.930:A=A+:G.540
535 GOS.970
540 N.O:E=E+4:D=-8:N.P
545 E=E+1:IFE?3<>I.(E/3)G.545
550 E=E/3*64
560 P.A.E;"HOW MANY PARTS (OR BOXES) ARE THERE ";:I.
561 IFP<>BGOS.799:G.560
565 P.A.E:REST.:F.X=1T.B:REA~A$:N.X
570 P.A.E;"HOW MANY ";A$;" ARE NOT SHADED IN ";:I.P
575 IFP<>AGOS.799:G.57
590 P.A.E:P.A.E+1;A:F=E/64*3+4:F.X=1T.8:S.(X,F):N.x
595 P.A.E+128+1;B:IFB=AJ=1:A$="WHOLES":G.601
600 REST.:F.X=1T.J:REA.A$:N.X:Y=J:IJ<>1J=A/C
601 Z=0:D=-1:F.P=1T.BS.C:F.O=1T.C-1:W=1:IFP>AW=3
602 D=D+8:IFW=1S.(D,Z):S.(D,Z+2):R.(D-1,Z+1):R.(D+1,Z+1)
603 IFW=3F.X=0T.2S.(D,X+Z):N.X
604 N.O:Z=Z+4:D=-1:N.P
605 IFB<>AG.610
638 Z=0:F.P=1T.BS.C:IFP>1S.(0,Z-1):S.(C*8-2,Z-1)
607 IFP>1F.X=1T.C*8-3:R.(X,Z-2):NX
608 IF(P<B)*(P>1)F.X=1T.C*8-3:R.(X,Z):N.X
609 Z=Z+4:N.P
610 P.A.E+74;"HOW MANY ";A$;" DOES THAT EQUAL ";:I.P
620 IFP<>JP.A.E+74;"SORR, TRY AGAIN":F.G=1T.500:N.G:G.610
621 IFA=BY=1
624 P.Q.E+1;J;A.E+129;Y;
630 P.A.E+192;"WELL DONE !":F.G=1T.999:N.G
798 C.:P.A.896;:N.Q:.2
799 P.A.E;"SORRY, PLEASE COUNT AGAIN.":F.Y=1T.99y:N.Y:P.A.E:RET.
800 P.A.896;:IFW=1P."I GOT YOU !":Q=Q+1
801 IFW=3P."AH !!!  YOU GT ME !":V=V+1
802 F.X=1T.500:N.X
805 IFW=3T=T-1
806 P.An896;"HITS:";V;" MISSES:";U;" SHOTS LEFT:";T;
807 IFQ>0P."   I'VE SHOT YOU";Q;"TME";:IFQ>1P."S";
808 P.".";:G.815
810 IFW=3U=U+1
811 G.x05
815 IFT<>0GOS.980:RET.
816 C.:P.A.460;:IFQ>VP."I WIN !   ";Q;"HIT";:IFQ>1P."S"
817 IFQ>VP." TO";V:G.821
818 IFV>QP."YOU WIN !  ";V;2HIT";:IFV>1P."S";
819 IFV>QP." TO";Q
820 IFQ=VP."WE CAME A DRAW !
821 F.X=1T.3000:NX:G.1
930 S.(A(W),A(2)+1)
940 F.X=A(W)T.A(W)+6:S.(X,A(2)):S.(X,A(2)+2):N.X
941 S.(A(W)+6,A(2)+1):RET.
950 F.X=A(W)T.A(W)+6:F.Y=A(W+1)T.(W+1)+2
960 R.(X,Y):N.Y:N.X:RET.
970 F.X=A(W)T.A(W)+6jF.Y=A(W+1)T.A(W+1)+2
975 S.(X,Y):N.Y:N.X:RET.
980 P.A.25;"< ROVING TARGETS >";
985 .A.1;:F.X=0T.7:S.(X,0):N.X:RET.
```

***** SOLITAIRE L2/ 16K by T. Griffin *****

This is the well known game for one player involving a board full of pegs but with one vacant space. The board is drawn on the screen as shown below and the player is requested to enter the number of the peg he wishes to move and the position to which it is to be moved. The computer ensures that all the rules are followed.

The program listing is well documented. Of particular note are lines 18Ø and 19Ø. Here, the identification numbers for each hole are printed by POKEing them directly into their correct position in screen memory. This avoids the leading space which would have occurred had the numbers been printed as variables and makes for a compact board layout.

```
20 DEFINTA-Z:GOTO540
30 CLS:CLEAR1000
40 '
******** ARRAYS AND DATA FOR VALIDITY CHECKS, AND POSITIONS FOR
NUMBERS AND PEG MARKERS

50 DIMV(33),C(33),P(33),N(33)
60 DATA13,14,15,22,23,24,29,30,31,32,33,34,35,38,39,40,41,42,43,
44,47,48,49,50,51,52,53,58,59,60,67,68,69
70 DATA148,154,160,276,282,288,392,398,404,410,416,422,428,520,5
26,532,538,544,550,556,648,654,660,666,672,678,684,788,794,800,91
6,922,928
80 DATA82,88,94,210,216,222,326,332,338,344,350,356,362,454,460,
466,472,478,484,490,582,588,594,600,606,612,618,722,728,734,850,8
56,862
90 FORA=1TO33:READV(A):NEXTA
100 FORA=1TO33:READP(A):NEXTA
110 '
******** BOARD IS PRINTED HERE

120 AA$=CHR$(170):AB$=CHR$(131):AC$=CHR$(171):AD$=CHR$(149):AE$=
CHR$(129):AF$=CHR$(130):AG$=CHR$(128):AH$=CHR$(143):Y$=AH$+AH$:N$
=AG$+AG$:D$=STRING$(4,AB$)+AC$+AD$:E$=AG$+Y$+AG$+AA$+AD$
130 D3$=AA$+AD$+D$+D$+D$:D7$=AA$+AD$+D$+D$+D$+D$+D$+D$+D$:E3$=AA
$+AD$+E$+E$+E$:E7$=AA$+AD$+E$+E$+E$+E$+E$+E$+E$:B2$=AF$+STRING$(1
1,AB$)+AC$+AD$+D$+D$+STRING$(4,AB$)+AC$+CHR$(151)+STRING$(11,AB$)
+AE$
140 PRINT@19,"S O L I T A I R E";:PRINT@81,D3$;:PRINT@145,E3$;:P
RINT@209,D3$;:PRINT@273,E3$;
150 PRINT@325,D7$;:PRINT@389,E7$;:PRINT@453,D7$;:PRINT@517,E7$;:
PRINT@581,D7$;:PRINT@645,E7$;
160 PRINT@709,B2$;:PRINT@785,E3$;:PRINT@849,D3$;:PRINT@913,E3$;:
PRINT@977,AF$+STRING$(18,AB$)+AE$;
170 '
******** BOARD NUMBERS INSERTED HERE. NOTE....POKEING THEM STOPS
 SPACES BEING INSERTED EITHER SIDE OF THE NUMBER.

180 FORX=1TO33:READN(X):X$=STR$(X):IFX<10POKE(N(X)+15360),X+48:N
EXT
190 X1=INT(X/10):POKE(N(X)+15360),X1+48:X2=X1*10:X3=X-X2:POKE(N(
X)+15361),X3+48:NEXT
200 '
******** MIDDLE PEG REMOVED

210 PRINT@P(17),N$;:C(17)=-1
220 '
******** "FROM?" AND "TO?" DISPLAYED

230 PRINT@372,"           ";:PRINT@500,"           ";:PRINT@372,"F
ROM?";:C=0:Z=378:GOSUB280
240 F=N
250 PRINT@500,"TO?";:C=0:Z=504:GOSUB280
260 T=N:GOSUB360:GOSUB440:GOSUB470:GOTO230
270 '
******** MOVES ENTERED HERE BY INKEY$, AND NUMBERS PROCESSED,
PRINTED, AND RETURNED FOR VALIDATION

280 A$=INKEY$:IFA$=""THEN280
290 IFA$="F"THEN490
300 IFA$=CHR$(13)THENRETURN
310 IFASC(A$)>47ANDASC(A$)<58THENN1=ASC(A$)-48:C=C+1
320 IFC=1THENN=N1
330 IFC=2THENN=N1+(N*10)
340 PRINT@Z,N;:GOTO280
350 '
******** VALIDATION OF REQUESTED MOVE DONE HERE

360 IFF<1ORF>33ORT<1ORT>33THEN420
```

```
370 F1=V(F):T1=V(T):J=ABS(F1-T1)
380 IFJ=2ORJ=18THEN390ELSE420
390 J1=(T1+F1)/2
400 FORJ2=1TO33:IFV(J2)=J1THEN410ELSENEXT
410 IFC(J2)=-1ORC(F)=-1ORC(T)=0THEN420ELSE RETURN
420 C=0:GOTO230
430 '
******** UPDATE OF BOARD AND GAME MEMORY (ARRAY C)

440 PRINT@P(F),N$;:PRINT@P(T),Y$;:PRINT@P(J2),N$;
450 C(F)=-1:C(T)=0:C(J2)=-1:S=S+1:RETURN
460 '
******** SCORE AND WIN MESSAGES

470 S1=32-S:IFS1=1THENPRINT@627,"AT LAST!!!!!";:PRINT@691,"YOU W
IN!!!!!";:ELSEPRINT@627,S1;"REMAINING   ";:RETURN
480 '
******** GAME RESTART AND INTRODUCTION

490 PRINT@756,"PLAY AGAIN?";
500 A$=INKEY$:IFA$=""THEN500
510 IFA$="Y"THENRUN
520 IFA$="N"THENEND
530 GOTO500
540 CLS:PRINTTAB(21);"S O L I T A I R E";
550 PRINT@320,"THIS IS A STANDARD GAME OF SOLITAIRE.  TO MOVE A
PIECE, ENTER"
560 PRINT:PRINT" THE NUMBER OF IT'S SQUARE, AND THE NUMBER OF TH
E SQUARE YOU"
570 PRINT:PRINT"       WISH TO MOVE TO.  TO FINISH THE GAME, ENTE
R 'F'."
580 PRINT:PRINT"              TO START THE GAME, PRESS ANY KEY."
590 A$=INKEY$:IFA$=""THEN590ELSE30
```

***** SEMAPHORE L2/4K by I.K. McAllister *****

This program is a fun way for boy scouts etc. to learn the semaphore system of signalling with hand flags. The program will just fit into 4K if the REM statements are omitted from the beginning. There are three levels:

Level 1 displays the signal with the signaller facing away - i.e. his right hand is your right hand, so you can copy the signal easily.

Intermediate has the signaller facing you, as would be the case when you receive a message.

In advanced level you may be required to interpret a signal received, or to build up a signal being sent.

To learn, I suggest taking 5 letters at a time in level 1 (6 for the last group) at speed 9. When you fail to name a signal the correct answer will be displayed for you to memorise. Having learned all the alphabet, select A to Z and gradually increase the speed to 1. When you find it easy to get a high score at speed 1 (and no longer confuse J with P) then select level 2, A to Z, speed 9, and see how you go with the mirror images.

When you can handle speed 1 in both of these levels, go on to advanced level. Speed 1 may be impossible for level 3, but you should reach speed 2. Remember an arm crossed over your chest is always held lower than the other arm. Remember that the signaller faces you when you receive, but away from you when you are "telling him" where to point his flags.

Line 3Ø sets up a table of equivalents for arm positions and letters of the alphabet.
Line 1ØØ avoids a nonsense letter being selected by the RND function, e.g. if you select letters Z to A the program will only use the letter A.
Lines 1ØØ to 12Ø are the main control parts of the program. The rest is made up of subroutines, mostly for the graphics.

One last twist: in level 3, any direction entered between the end of right hand input and the input of the left hand is accepted as a left hand input. This is useful if you enter R & L hands in quick succession. However, if you enter R.H. just too late, it is counted as left hand, so you lose that point too. If you dislike this, then use EDIT 24Ø to insert :C$=INKEY$ immediately after the " mark following the words LEFT HAND.

```
20 CLEAR5
30 DIMA$(26),R(26),L(26)
40 DATAA,1,8,B,2,8,C,3,8,D,4,8,E,8,5,F,8,6,G,8,7,H,2,1,I,3,1,J,4
,6,K,1,4,L,1,5,M,1,6,N,1,7,O,3,2,P,2,4,Q,2,5,R,2,6,S,2,7,T,3,4,U,
3,5,V,4,7,W,6,5,X,7,5,Y,3,6,Z,7,6
```

```
50 FORA=1TO26:READA$(A):READR(A):READL(A):NEXT
60 CLS
70 PRINT"WHAT LEVEL ARE YOU?":INPUT"1=BEGINNER,2=INTERMEDIATE,3=
ADVANCED";K:IFK>3THENPRINT"WHAT?!!!":GOTO70
80 PRINT"BETWEEN WHICH LETTERS DO YOU WANT TO PRACTICE? 1ST LETT
ER";:L=1:GOSUB340:PRINT"LAST";:L=2:GOSUB340
90 INPUT"HOW FAST(1TO9)--1 IS FASTEST";G:J=K:IFG>9THEN90ELSEIFK=
3THENJ=2
100 CLS:PRINT"THE LETTER ";A$(A);" IS";:GOSUB500:GOSUB540:S=0:CL
S
110 H=N(2)-N(1)+1:A=RND(H)-1+N(1):IFA<1THEN110ELSEONKGOTO120,130
,140
120 J=1:GOTO150
130 J=2:GOTO150
140 L=RND(2):IFL=2GOTO230ELSEJ=2
150 PRINT@1,"WHAT LETTER IS THIS?":GOSUB500
160 FORP=1TO125*G:C$=INKEY$:IFC$=""THENNEXT
170 IFC$=A$(A)THENS=S+2:GOTO190
180 PRINT@1,CHR$(30):PRINT@1,"THE LETTER IS ";A$(A):FORP=1TO2500
:NEXT:S=S-2
190 CLS:PRINT@896,"YOUR SCORE IS ";S;:PRINT@1,"TO END HIT ANY KE
Y"
200 C$=INKEY$:FORP=1TO125:C$=INKEY$:IFC$<>""THEN220
210 NEXT:GOTO110
220 CLS:PRINTCHR$(23):PRINT"YOUR FINAL SCORE WAS";S;:END
230 CLS:GOSUB490:PRINT@1,"**LETTER ";A$(A);"**IN WHICH DIRECTION
 SHOULD YOU AIM YOUR RIGHT HAND?":FORP=1TO125*G:C$=INKEY$:IFC$=""
THENNEXT
240 F=R(A):R(A)=VAL(C$):GOSUB520:R(A)=F:IFR(A)=VAL(C$)THENS=S+1E
LSECLS:PRINT@1,"WRONG--THE RIGHT ONE IS":GOSUB490:GOSUB520:S=S-1
250 CLS:GOSUB490:PRINT@1,"**LETTER ";A$(A);"**IN WHICH DIRECTION
 SHOULD YOU AIM YOUR LEFT HAND?":FORP=1TO125*G:C$=INKEY$:IFC$=""T
HENNEXT
260 F=L(A):L(A)=VAL(C$):GOSUB530:L(A)=F:IFL(A)=VAL(C$)THENS=S+1E
LSECLS:PRINT@1,"WRONG--THE RIGHT ONE IS":GOSUB490:GOSUB530:S=S-1
270 CLS:GOTO190
280 FORX=35TO39:SET(X,14):SET(X,19):NEXT:SET(34,15):SET(40,15):S
ET(34,18):SET(40,18):FORY=16TO17:SET(33,Y):SET(41,Y):NEXT:FORY=15
TO17:SET(32,Y):SET(42,Y):NEXT
290 FORY=20TO26:SET(37,Y):NEXT:X=37:FORY=26TO31:X=X-1:SET(X,Y):N
EXT:X=37:FORY=26TO31:X=X+1:SET(X,Y):NEXT:SET(30,31):SET(44,31):FO
RX=35TO39:SET(X,21):NEXT:RETURN
300 X=39:FORY=22TO27:X=X+2:SET(X,Y):NEXT:X=51:Y=27:GOSUB470:RETU
RN
310 FORX=40TO61:SET(X,21):NEXT:X=54:Y=21:GOSUB480:RETURN
320 X=57:FORY=13TO20:X=X-2:SET(X,Y):NEXT:X=55:Y=13:GOSUB470:RETU
RN
330 FORY=10TO20:SET(40,Y):NEXT:X=40:Y=10:GOSUB480:RETURN
340 INPUTC$:FORA=1TO26:IFC$=A$(A)THENN(L)=AELSENEXT
350 RETURN
360 FORX=21TO34:SET(X,21):NEXT:X=21:Y=21:GOSUB480:RETURN
370 X=40:FORY=22TO27:X=X-2:SET(X,Y):NEXT:X=28:Y=27:GOSUB470:RETU
RN
380 FORY=22TO31:SET(40,Y):NEXT:RETURN
390 X=35:FORY=22TO27:X=X-2:SET(X,Y):NEXT:X=23:Y=27:GOSUB470:RETU
RN
400 FORX=21TO34:SET(X,21):NEXT:X=13:Y=21:GOSUB480:RETURN
410 X=17:FORY=13TO20:X=X+2:SET(X,Y):NEXT:X=19:Y=13:GOSUB470:RETU
RN
420 FORY=10TO20:SET(34,Y):NEXT:X=27:Y=10:GOSUB480:RETURN
430 PRINT"NEVER CROSS ARMS UPWARD":GOSUB540:RETURN
440 FORX=40TO45:SET(X,21):NEXT:X=46:Y=21:GOSUB480:RETURN
450 X=34:FORY=22TO27:X=X+2:SET(X,Y):NEXT:X=46:Y=27:GOSUB470:RETU
RN
460 FORY=22TO31:SET(34,Y):NEXT:RETURN
470 FORD=(X-2)TO(X+2):SET(D,Y+1):SET(D,Y+3):NEXT:FORD=(X-4)TO(X+
4):SET(D,Y+2):NEXT:SET(X,Y+4):RETURN
480 FORD=XTO(X+7):FORZ=YTO(Y+3):SET(D,Z):NEXTZ:NEXTD:RETURN
490 GOSUB280:PRINT@668,"1";:PRINT@481,"2";:PRINT@220,"3";:PRINT@
146,"4";:PRINT@202,"5";:PRINT@451,"6";:PRINT@648,"7";:PRINT@722,"
8";:RETURN
500 GOSUB280:IFJ=2THENSET(36,16):SET(38,16):ONR(A)GOSUB390,400,4
10,420,430,440,450,460:ONL(A)GOSUB370,360,340,330,320,310,300,380
:RETURN
510 GOSUB520:GOTO530
520 ONR(A)GOSUB300,310,320,330,430,360,370,380:RETURN
530 ONL(A)GOSUB450,440,430,420,410,400,390,460:RETURN
540 FORP=1TO2000:NEXT:RETURN
```

***** THREE BILLY GOATS GRUFF L2/16K by R. Norman *****

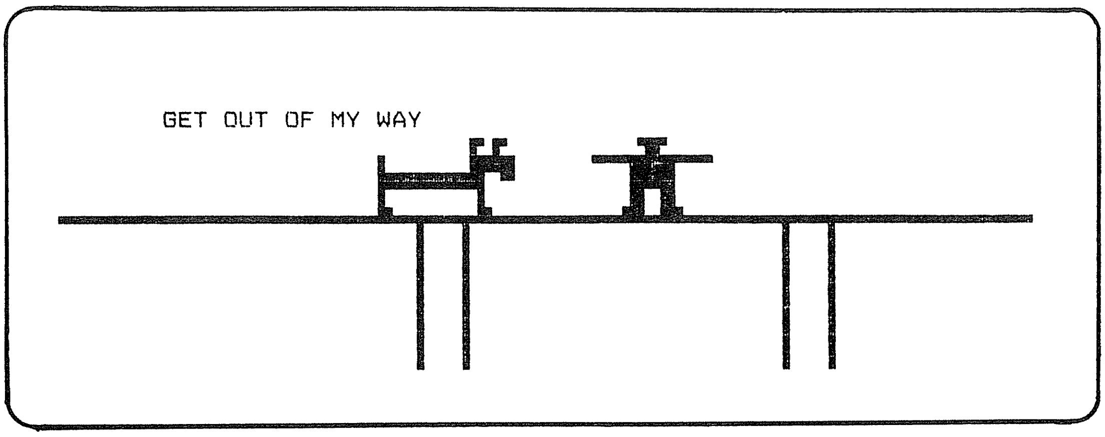
This Basic program runs on a Level II 16K '80 and requires less than 6K of memory. Simply CLOAD and RUN, then press the space bar (or any other key) to turn to the next page of the story.

Our justification (rationalisation) for buying a computer was that we wanted our children to grow up thinking that computers are part of the furniture, and this program is part of an attempt to make our children more interested in that item of furniture.

The program is an animated presentation of a well known children's tale. I chose this tale for conversion to a program both because the story was my three year old son's favourite at the time and because, with few characters and very little scenery, the graphics for the story are relatively easy.

The text is very basic: it is a simplified version of a Ladybird book for beginner readers (5-6 year olds). However, the program also appeals to those who are much younger (my 15 month old daughter enjoys it thoroughly, though insists that the goats are dogs), and most adults have found it entertaining.

From a programming point of view, the most interesting aspect of the program is the use of control codes 26 and 24 within graphics characters. Control code 26 moves the cursor down one line; control code 24 moves the cursor back one space. The use of these codes with the ordinary graphics codes (128-191) permits the creation of a single graphics character occupying, say, three video lines that can be moved around the screen as if it were a single letter or a single-line string, provided that the boundaries of the screen are respected. An example of this can be found in lines 41Ø-44Ø, which use the data in lines 16Ø-2ØØ to create a middle-sized goat (M$) that takes up three lines of the screen. In lines 1Ø7Ø and 123Ø this string is moved as if it were a single-line string. Note also the use of the graphics character '128', the blank character. It is necessary to place these characters on the side that the overall graphics figure (e.g. the goat) is coming from in order to prevent a trail of blobs being left behind. For example, the goats move from left to right in the story, so that they must have blank characters on the left side. (Note that the troll, which moves up and down, is an exception. It has blank characters on top to cover the case where the troll descends, but I found that I could not place the characters under the troll because they destroyed, and would not permit the reconstruction of, the bridge while the troll was supposed to be standing on it. To cover this case I had to use a separate variable, TC$, under the troll for the time that the troll was moving upwards - see lines 1ØØ and 18ØØ.



```
80 CLS: CLEAR 1000
90 PRINTCHR$(23):PRINT@390,"THREE BILLY GOATS GRUFF"
100 C$=STRING$(64,128):C2$=C$+C$:C3$=C$+C$+C$:TC$=STRING$(4,128)
:G$=CHR$(164)+CHR$(184)
110 REM LITTLE GOAT
120 DATA 128,144,128,128,128,170,172
130 DATA 26,24,24,24,24,24,24,24
140 DATA 128,183,131,131,131,171,144
150 REM MIDDLE SIZED GOAT
160 DATA 128,128,128,128,128,144
170 DATA 26,24,24,24,24,24,24,24
180 DATA 128, 180,176,176,176,176,191,139
190 DATA 26,24,24,24,24,24,24,24,24
200 DATA 128, 183,131,131,131,131,171,144
210 REM LARGE GOAT
220 DATA 128,144,128,128,128,128,128,183,186,177
230 DATA 26,24,24,24,24,24,24,24,24,24,24
240 DATA 128, 189,188,188,188,188,188,191,131,143
```

DON'T BE HELD BACK BY AN ANTIQUATED DISK OPERATING SYSTEM

MOVE UP TO

NEWDOS 80 **$149 incl. p&p**

NEWDOS 80 is a completely new DOS for the TRS-80 SYSTEM 80. It is well-documented, bug free and increases the power of your system many times over. It is upward compatible with TRSDOS AND NEWDOS (ie TRSDOS and NEWDOS+ programs will run on NEWDOS 80 but the reverse is not necessarily so).

These are just a few of the many new features offered by NEWDOS 80.

* New BASIC commands that support variable record lengths up to 4095 bytes long.
* Mix or match disk drives. Supports any track count from 18 to 96. Use 35, 40, 77 or 80 track 5¼ inch mini disk drives, 8 inch disk drives OR ANY COMBINATION.
* An optional security boot-up for BASIC or machine code application programs. User never sees "DOS-READY" or "READY" and is unable to "BREAK", clear screen or issue any direct BASIC statements, including "LIST".
* New editing commands that allow program lines to be deleted from one location and moved to another or to allow the duplication of a program line with the deletion of the original.
* Enhanced and improved RENUMBER that allows relocation of subroutines.
* Create powerful chain command files which will control the operation of your system.
* Device handling for routing to display and printer simultaneously.
* MINIDOS – striking the D, F and G keys simultaneously calls up a MINIDOS which allows you to perform many of the DOS commands without disturbing the resident program.
* Includes Superzap 3.0 which enables you to display/print/modify any byte in memory or on disk.
* Also includes the following utilities:
  - Disk Editor/Assembler
  - Disassembler (Z80 machine code)
  - LM offset – allows transfers of any system tape to Disk file – automatically relocated.
  - LEVEL I – Lets you convert your computer back to Level 1.
  - LVIDKSL – Saves and loads Level 1 programs to disk.
  - DIRCHECK – Tests disk directories for errors and lists them.
  - ASPOOL – An automatic spooler which routes a disk file to the printer whilst the computer continues to operate on other programs.
  - LCDVR – a lower case drives which display lower case on the screen if you have fitted a simple lower case modification.

DISK DRIVE USERS ELIMINATE CRC ERRORS AND TRACK LOCKED OUT MESSAGES FIT A PERCOM DATA SEPARATOR

$37.00 plus $1.20 p&p.

When Tandy designed the TRS-80 expansion interface, they did not include a data separator in the disk-controller circuitry, despite the I.C. manufacturer's recommendations to do so. The result is that many disk drive owners suffer a lot of Disk I/O errors. The answer is a data separator. This unit fits inside your expansion interface. It is supplied with full instructions and is a must for the serious disk user.

MPI DISK DRIVES HIGHER PERFORMANCE – LOWER PRICE

MPI is the second largest manufacturer of disk drives in the world. MPI drives use the same form of head control as 8" drives and consequently, they have the fastest track-to-track access time available – 5msec! All MPI drives are capable of single or double-density operation. Double-density operation requires the installation of a PERCOM doubler board in the expansion interface.

As well as single head drives, MPI also makes dual-head drives. A dual-head drive is almost as versatile as two single-head drives but is much cheaper.

Our MPI drives are supplied bare or in a metal cabinet – set up to operate with your TRS-80 or SYSTEM 80. All drives are sold with a 90 day warranty and service is available through MICRO-80 PRODUCTS.

MPI B51 40 Track Single Head Drive. . . . . .only $349
MPI B52 40 Track Double Head Drive. . . . .only $449

Prices are for bare drives and include p&p. Add $10.00 per drive for a cabinet and $60.00 for a power supply to suit two drives. 40 track drives are entirely compatible with 35 track drives. A 40 track DOS such as NEWDOS 80 is necessary to utilise the extra 5 tracks.

OVER 800 KILOBYTES ON ONE DISKETTE! WITH MPI 80 TRACK DRIVES

MPI 80 track drives are now available. The B91 80 track single-head drive stores 204 Kilobytes of formatted data on one side of a 5¼ inch diskette in single-density mode. In double-density mode it stores 408 Kilobytes and loads/saves data twice as quickly.

The B92 80 track dual-head drive stores 204 Kilobytes of formatted data on EACH side of a 5¼ inch diskette in single-density mode. That's 408 Kilobytes per diskette. In double-density mode, the B92 stores a mammoth 408 Kilobytes per side or 816 Kilobytes of formatted data per diskette. With two B92's and a PERCOM double, you could have over 1.6 Megabytes of on line storage for your TRS-80 for less than $1500!!

MPI B91 80 Track Single Head Drive. . . . . .only $499
MPI B92 80 Track Dual Head Drive . . . . . .only $619

Prices are for bare drives and include p&p. Add $10.00 per drive for a cabinet and $60.00 for a power supply to suit two drives. Note: 80 track drives will not read diskettes written on a 35 or 40 track drive. If drives with different track counts are to be operated on the same system, NEWDOS 80 must be used.

CARE FOR YOUR DISK DRIVES? THEN USE 3M's DISK DRIVE HEAD CLEANING DISKETTES

$30.20 incl. p&p.

Disk drives are expensive and so are diskettes. As with any magnetic recording device, a disk drive works better and lasts longer if the head is cleaned regularly. In the past, the problem has been, how do you clean the head without pulling the mechanism apart and running the risk of damaging delicate parts. 3M's have come to our rescue with SCOTCH BRAND, non-abrasive, head cleaning diskettes which thoroughly clean the head in seconds. The cleaning action is less abrasive than an ordinary diskette and no residue is left behind. Each kit contains:

- 2 head cleaning diskettes
- 1 bottle of cleaning fluid
- 1 bottle dispenser cap

PLEASE USE ORDER FORM ON PAGE 36

USE TANDY PERIPHERALS ON YOUR SYSTEM-80 VIA
SYSPAND-80 - $97.50 incl. p&p

The SYSTEM-80 hardware is not compatible with the TRS-80 in two important areas. The printer port is addressed differently and the expansion bus is entirely different. This means that SYSTEM-80 owners are denied the wealth of economical, high performance peripherals which have been developed for the TRS-80. Until now, that is. MICRO-80 has developed the SYSPAND-80 adaptor to overcome this problem. A completely self-contained unit in a small cabinet which matches the colour scheme of your computer, it connects to the 50-way expansion part on the rear of your SYSTEM 80 and generates the FULL Tandy 40 way bus as well as providing a Centronics parallel printer port. SYSPAND-80 enables you to run an Exatron Stringy Floppy from your SYSTEM 80, or an LNW Research expansion interface or any other desirable peripherals designed to interface to the TRS-80 expansion port. Make your SYSTEM 80 hardware compatible with the TRS-80 via SYSPAND-80.

PROGRAMS BY MICROSOFT

EDITOR ASSEMBLER PLUS (L2/16K)
$37.50 + $1.20 p&p

A much improved editor-assembler and debug/monitor for L2/16K TRS-80 or SYSTEM 80. Assembles directly into memory, supports macros and conditional assembly, includes new commands-substitute, move, copy and extend.

LEVEL III BASIC **$59.95 plus $1.20 p&p**

Loads on top of Level II BASIC and gives advanced graphics, automatic renumbering, single stroke instructions (shift-key entries) keyboard debounce, suitable for L2/16K and up (Not Disk BASIC)

ADVENTURE ON DISK **$35.95 plus $1.20 p&p**

This is the original ADVENTURE game adapted for the TRS-80. The game fills an entire diskette. Endless variety and challenge as you seek to rise to the level of Grand Master. Until you gain skill, there are whole areas of the cave that you cannot enter. (Requires 32K One Disk)

BASIC COMPILER **$208 plus $2.00 p&p**

New improved version, the Basic Compiler converts Disk BASIC programs to machine code, automatically. A compiled program runs, on average, 3-10 times faster than the original BASIC program and is much more difficult to pirate.

UPGRADE TO 16K
FOR ONLY $30.00!!

MICRO-80's 16K MEMORY EXPANSION KIT HAS BEEN REDUCED IN PRICE EVEN MORE

Larger volume means we buy better and we pass the savings on to you. These are our proven, prime, branded 200 ns (yes, 200 nanosecond) chips. You will pay much more elsewhere for slow, 350 ns. chips. Ours are guaranteed for 12 months. A pair of DIP shunts is also required to upgrade the CPU memory in the TRS-80 — these cost an additional $4.00. All kits come complete with full, step-by-step instructions which include labelled photographs. No soldering is required. You do not have to be an experienced electronic technician to instal them.

DISK DRIVE CABLES
SUITABLE FOR ANY DISK DRIVES

DC-2 2 Drive Connector Cable	$39 incl. p&p
DC-4 4 Drive Connector Cable	$49 incl. p&p

DOUBLE THE SPEED AND CAPACITY OF YOUR DISK DRIVES
PERCOM DOUBLER ONLY $220
plus $2.00 p&p

Installing a Doubler is like buying another set of disk drives, only much cheaper!! The doubler works with most modern disk drives including:- MPI, Micropolis, Pertec, TEAC (as supplied by Tandy). The doubler installs in the TRS-80 expansion interface, the System-80 expansion interface and the LNW Research expansion interface in a few minutes without any soldering, cutting of tracks, etc. It comes complete with its own TRSDOS compatible double density operating system.

DOUBLE-ZAP II — DOUBLE DENSITY PATCH FOR NEWDOS 80
ONLY $53.00 plus $1.00 p&p

DISCONTINUED

If you are using NEWDOS 80, then you also need DOUBLE-ZAP II, on diskette. This program upgrades your NEWDOS 80 to double density with ADR (automatic density recognition). It retains all the familiar features, including the ability to mix and match track counts on the same cable. In addition, it gives NEWDOS 80 the ability to mix densities on the same cable, automatically. If you place a single density diskette in drive Ø, say and a double density diskette in drive 1, Double-ZapII will recognise this and read/write to drive Ø in single density whilst at the same time it reads/writes to drive 1 in double density!

FLOPPY DOCTOR AND MEMORY DIAGNOSTIC
(by MICRO CLINIC) $29.95 plus 50c. p&p

Two machine language programs on a diskette together with manual which thoroughly test your disk drives and memory. There are 19 possible error messages in the disk drive test and their likely causes are explained in the manual. Each pass of the memory tests checks every address in RAM 520 times, including the space normally occupied by the diagnostic program itself. When an error occurs the address, expected data, and actual data are printed out together with a detailed error analysis showing the failing bit or bits, the corresponding IC's and their location. This is the most thorough test routine available for TRS-80 disk users.

BOOKS

LEVEL II ROM REFERENCE MANUAL
$24.95 + $1.20 p&p

Over 70 pages packed full of useful information and sample programs. Applies to both TRS-80 and SYSTEM 80.

TRS-80 DISK AND OTHER MYSTERIES
$24.95 + $1.20 p&p

The hottest selling TRS-80 book in the U.S.A. Disk file structures revealed, DOS's compared and explained, how to recover lost files, how to rebuild crashed directories — this is a must for the serious Disk user and is a perfect companion to any of the NEWDOS's.

LEARNING LEVEL II
$16.95 + $1.20 p&p

Written by Daniel Lien, the author of the TRS-80 Level I Handbook, this book teaches you, step-by-step, how to get the most from your Level II machine. Invaluable supplement to either the TRS-80 Level II Manual or the System-80 Manuals.

MORE AUSTRALIAN SOFTWARE

All programs designed to run on both the TRS-80 or the SYSTEM 80 without modification. Most programs include sound

TRIAD VOL 1 – L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

Three separate games which test your powers of memory and concentration. The programs combine graphic displays and sound:

SIMON-SEZ: Just like the electronic music puzzles on sale for more than $20. Numbers are flashed on the screen and sounded in a sequence determined by the computer. Your task is to reproduce the sequence, correctly.

LINE?: Rather like a super, complicated version of noughts and crosses. You may play against another player or against the computer itself. But beware, the computer cheats!

SUPER CONCENTRATION: Just like the card game but with more options. You must find the hidden pairs. You may play against other people, play against the computer, play on your own, or even let the '80 play on its own.

TRIAD VOL 2 – L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

Remember those "NUMERO" puzzles in which you had a matrix of numbers (or letters) with one blank space and you had to shuffle the numbers around one at a time until you had made a particular pattern? Well, **SHUFFLEBOARD**, the first program in this triad, is just this, except that the computer counts the number of moves you take to match the pattern it has generated — so it is not possible to cheat.

MIMIC is just like SHUFFLEBOARD except that you only see the computer's pattern for a brief span at the beginning of the game, then you must remember it!

In **MATCHEM**, you have to manoeuvre 20 pegs from the centre of the screen to their respective holes in the top or bottom rows. Your score is determined by the time taken to select a peg, the route taken from the centre of the screen to the hole and your ability to direct the peg into the hole without hitting any other peg or the boundary.

VISURAMA L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

Two programs which give fascinating, ever-changing patterns on the screen.

LIFE is the fastest implementation of the Game of Life you will see on your '80. Machine language routines create up to 1200 new generations per minute for small patterns or up to 100 per minute for the full 128 x 48 screen matrix. Features full horizontal and vertical wraparound.

EPICYCLES will fascinate you for hours. The ever-changing ever-moving patterns give a 3D effect and were inspired by the ancient Greek theories of Ptolemy and his model of the Solar system.

EDUCATION AND FUN – L1/4K, L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

Written by a primary school teacher to make learning enjoyable for his pupils, there are five programs in both Level I and Level II to suit all systems:

BUG-A-LUG: a mathematics game, in which you must get the sum correct before you can move.

AUSTRALIAN GEOGRAPHY: learn about Australian States and towns, etc.

SUBTRACTION GAME: build a tower with correct answers.

HOW GOOD IS YOUR MATHS? Select the function (+, –, ÷ or X) and degree of difficulty.

HANGMAN: That well known word game now on your computer.

Recommended for children from 6 to 9 years.

COSMIC FIGHTER & SPACE JUNK – L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

Both programs have sound to complement their excellent graphics. In **COSMIC FIGHTER**, you must defend the earth against seven different types of alien aircraft. It is unlikely that you will be successful but you will have a lot of fun trying!

You mission in **SPACE JUNK** is to clean up all the debris left floating around in space by those other space games. It is not as simple as it sounds and space junk can be quite dangerous unless you are very careful.

SPACE DRIVE L2/4K & 16K
Cassette $8.95 Disk $13.95
+ 60c p&p

Try to manoeuvre your space ship through the meteor storms then land it carefully at the space port without running out of fuel or crashing. Complete with realistic graphics.

STARFIRE AND NOVA INVASION L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

Both programs include sound to improve their realism.

STARFIRE seats you in the cockpit of an X-wing fighter as you engage in battle with the deadly Darth Vader's Tie-fighters. Beware of the evil one himself and may the Force be with you.

In **NOVA INVASION**, you must protect your home planet of Hiberna from the invading NOVADIANS. You have two fixed guns at each side of the screen and a moveable one at the bottom. Apart from shooting down as many invaders as possible, you must protect your precious hoard of Vitaminium or perish!

AIR ATTACK AND NAG RACE – L2/16K
Cassette $10.95 Disk $15.95
+ 60c p&p

An unlikely combination of programs but they share the same author who has a keen sense of humour.

AIR ATTACK includes sound and realistic graphics. The aircraft even have rotating propellors! But they also drop bombs on you, so it's kill or be killed!

NAG RACE lets you pander to your gambling instinct without actually losing real money. Up to five punters can join in the fun. Each race results in a photo-finish whilst there is a visible race commentary at the bottom of the screen throughout the race. Happy punting!

FOUR LETTER MASTERMIND L2/16K
Cassette $8.95 Disk $13.95
+ 60c p&p

There are 550 four-letter words from which the computer can make its choice. You have 12 chances to enter the correct word. After each try, the computer informs you of the number of correct letters and those in the correct position. You can peek at the list of possible words but it will cost you points. Makes learning to spell fun.

MUSIC IV – L2/16K
Cassette $8.95 Disk $13.95
+ 60c p&p

Music IV is a music compiler for your '80. It allows you to compose or reproduce music with your computer that will surprise you with its range and quality. You have control over duration (full beat to 1/16 beat) with modifications to extend the duration by half or one third for triplets. Both sharps and flats are catered for as are rests. Notes on whole sections may be repeated. The program comes with sample data for a well-known tune to illustrate how it is done.

SAVE 00$'s***SAVE 00$'sSAVE 00$'s***MICRO-80 EXPANSION INTERFACE***

MICRO-80's expansion interface utilises the proven LNW Research Expansion board. It is supplied fully built up and tested in an attractive cabinet with a self contained power supply, ready to plug in and go. The expansion interface carries MICRO-80's full, no hassle, 90-day warranty.

Features include:- ● Sockets for up to 32K of memory expansion ● Disk controller for up to 4 disk drives ● Parallel printer port ● Serial RS232C/20mA I/O port ● Second cassette (optional)

The expansion interface connects directly to your TRS-80 L2/16K keyboard or, via SYSPAND-80 to your SYSTEM-80VIDEO GENIE

Prices: HD-010-A Expansion Interfaces with Ø K : $499.00 HD-010-B Expansion Interfaces with 32K : $549.00 HD-011 Data separator fitted (recommended) : add $29.00 HD-012 Dual cassette Interfaces fitted : add $19.00

The MICRO-80 Expansion Interface is also available in kit form.

Prices: HD-013 Kit consisting of LNW Research PC board and manual, ALL components including cabinet & power supply : $375.00 HD-011 Data separator for above $25.00 HD-013 Dual cassette Interface kit : $15.00

TURN THIS

into this

for $49.00 plus $2.00 p & p

A choice of upper and lower case display is easier to read, gives greater versatility.

The Micro-80 lower case modification gives you this facility, plus the symbols for the 4 playing-card suits for $49.00 + $2.00 p. & p.

The Micro-80 modification features true below-the-line descenders and a block cursor.

Each kit comes with comprehensive fitting instructions and two universal lower-case drive routines on cassette to enable you to display lower case in BASIC programs.

The driver routines are self-relocating, self-protecting and will co-reside with other machine language programs such as Keyboard-debounce, serial interface driver programs etc.

Both programs give your TRS-80™ Model I or System 80™ an optional typewriter capability, i.e. shift for upper case.

The second programme also includes Keyboard-debounce and a flashing cursor.

You fit it. Or we can.

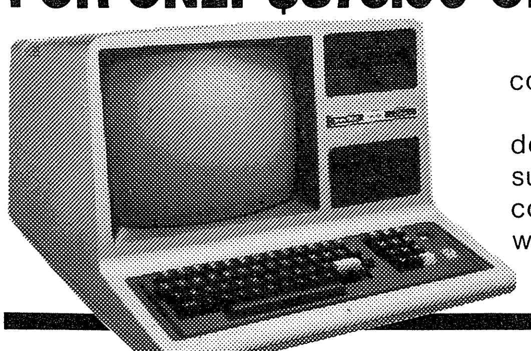
Fitting the modification requires soldering inside the computer. This should only be carried out by an experienced hobbyist or technician.

If you are at all dubious, a fitting service is available in all capital cities for only $20.00.

A list of installers is included with each kit.

Save $120 now.

ADD A DISK DRIVE TO YOUR TRS-80™ MODEL III FOR ONLY $875.00 OR ADD TWO FOR ONLY $1199.



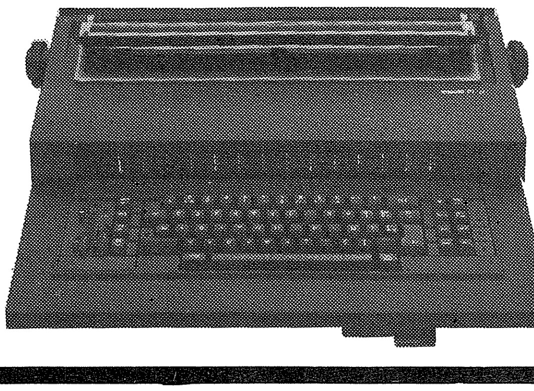
The Micro-80 disk drive upgrade for the TRS-80™ Model III contains the following high quality components:

1 or 2 MPI 40-track single head disk drives, 1 VR Data double-density disk controller board and 1 dual drive power supply plus all the necessary mounting hardware, cables and comprehensive fitting instructions, which can be carried out with a minimum of fuss by any average computer owner.

Fitting service is available for $25.00 in most capital cities.

ONLY $2049 INC. S.T.

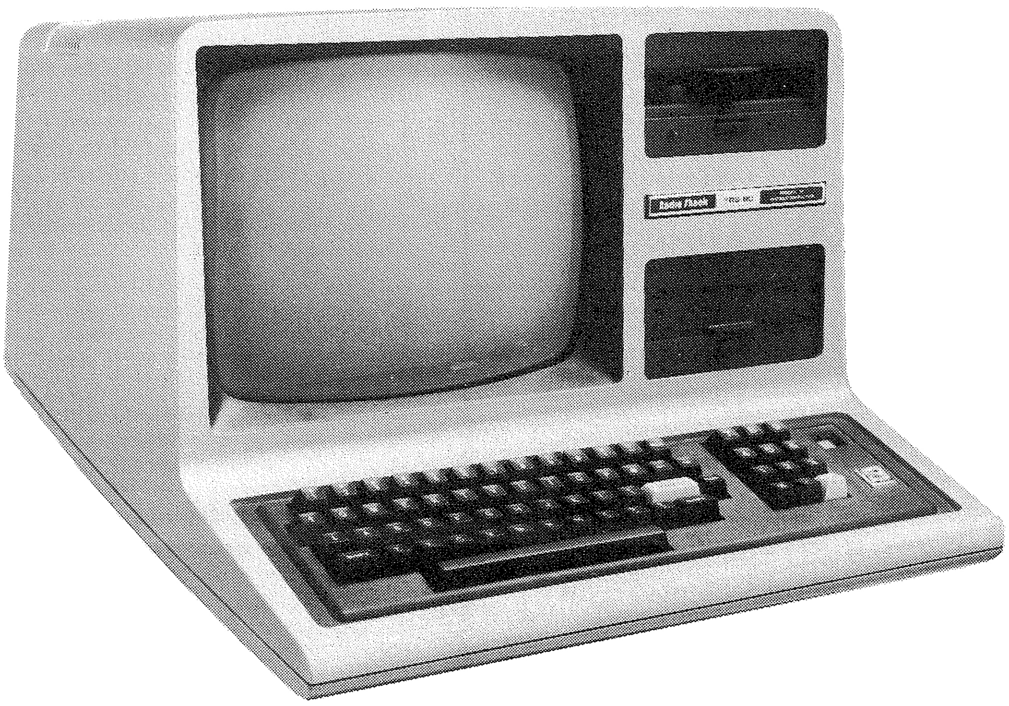
Daisy Wheel Typewriter/Printer



MICRO-80 has converted the new OLIVETTI ET-121 DAISY WHEEL typewriter to work with the TRS-80 and SYSTEM 80 or any other microcomputer with a Centronics parallel port (RS 232 serial interface available shortly). The ET-121 typewriter is renowned for its high quality, fast speed (17 c.p.s.), quietness and reliability. MICRO-80 is renowned for its knowledge of the TRS-80/SYSTEM 80 and its sensible pricing policy. Together, we have produced a dual-purpose machine:- an attractive, modern, correcting typewriter which doubles as a correspondence quality Daisy-wheel printer when used with your micro-computer.

How good is it? - This part of our advertisement was typeset using an ET-121 driven by a TRS-80. Write and ask for full details.

1.4 MEGABYTES ON LINE + 48K RAM for $3800 incl. Sales Tax



MICRO-80's

MODEL 380 +

MICRO-80 has equipped the TRS-80 with two high reliability dual-head 80 track mini-floppy disk drives made by MPI, one of America's leading mini-disk drive manufacturers.

This turns the mild-mannered Model 3 into a powerhouse able to handle the most difficult business programs. The TRS-80 is one of the best-supported microcomputers in the world. MICRO-80 has been supporting the TRS-80 in Australia for 18 months and is one of Australia's leading dealers in MPI disk drives.

2.8 MEGABYTES FOR $5300 incl. Sales Tax

If you need even more file space you can add MICRO-80's external dual-drive cabinet enclosing two more dual-head 80 track drives for an additional $1500.

COMPUTER PRICES

MODEL 340 2 40 TRACK SINGLE HEAD DRIVES GIVING 350K FORMATTED STORAGE, 48K RAM	$2990 INCL. SALES TAX
MODEL 340 + 2 40 TRACK DUAL-HEAD DRIVES GIVING 700K FORMATTED STORAGE, 48K RAM	$3350 INCL. SALES TAX
MODEL 380 2 80 TRACK SINGLE HEAD DRIVES GIVING 700K FORMATTED STORAGE, 48K RAM	$3350 INCL. SALES TAX
MODEL 380 + 2 80 TRACK DUAL-HEAD DRIVES GIVING 1.4 MEGABYTE FORMATTED STORAGE, 48K RAM	$3800 INCL. SALES TAX
350K SYSTEM MODEL 340, EPSON MX-80 PRINTER NEWDOS 80 DISK OPERATING SYSTEM	$4070 INCL. SALES TAX
700K SYSTEM (40 Track) MODEL 340 +, EPSON MX-80 PRINTER NEWDOS 80 DISK OPERATING SYSTEM	$4429 INCL. SALES TAX
700K SYSTEM (80 Track) MODEL 380, EPSON MX-80 PRINTER NEWDOS 80 DISK OPERATING SYSTEM	$4429 INCL. SALES TAX
1.4 MEGABYTE SYSTEM MODEL 380 +, EPSON MX-80 PRINTER NEWDOS 80 OPERATING SYSTEM	$4880 INCL. SALES TAX
2.8 MEGABYTE SYSTEM MODEL 380 +, DUAL EXTERNAL DRIVES, MX-80 PRINTER, NEWDOS 80 OPERATING SYSTEM	$6380 INCL. SALES TAX

★

EXATRON STRINGY FLOPPY — $372.50 Incl. P&P

All Exatron Stringy Floppies sold by MICRO-80 include the special chained version of **HOUSEHOLD ACCOUNTS,** developed by Charlie Bartlett. When used on the ESF, this program is powerful enough to perform many of the accounting functions in a small business. Remember, the ESF comes complete with a comprehensive manual, a 2 way bus-extender cable, its own power supply and 10 wafers of mixed length. One wafer contains the Data Input/Output program and another the **HOUSEHOLD ACCOUNTS** program.

CAN'T MAKE UP YOUR MIND ABOUT THE ESF?

Then send in $5.00 for a copy of the manual. We will refund your $5.00 IN FULL when you purchase an ESF.

★

All prices subject to change without notice. Prices are F.O.B. Adelaide. All computers and systems carry MICRO-80's 90-day Warranty covering parts and labour.

SOFTWARE BY AUSTRALIAN AUTHORS

All our software is suitable for either the SYSTEM 80 or the TRS-80

NEW SOFTWARE FROM MICRO-80 PRODUCTS

BUSINESS PROGRAMS

MICROMANAGEMENT STOCK RECORDING SYSTEM (L2/16K)

Cassette version. . . . . $29.95 + $1.00 p&p
Stringy Floppy version. . . . . $33.95 + $1.00 p&p

This system has been in use for 9 months in a number of small retail businesses in Adelaide. It is therefore thoroughly debugged and has been tailor made to suit the requirements of a small business. MICROMANAGEMENT SRC enables you to monitor the current stock level and reorder levels of 500 different stock items per tape or wafer. It includes the following features:–

- Add new items to inventory
- Delete discontinued items from inventory
- List complete file
- Search for any stock number
- Save data to cassette or wafer
- Load data from cassette or wafer
- Adjusts stock levels from sales results and receipt of goods
- List all items requiring reordering

We can thoroughly recommend this program for the small business with a L2/16K computer.

SCOTCH BRAND COMPUTING CASSETTES

Super-quality personal computing cassettes.

C-10 pack of 10	$26.00 incl. p&p
C-30 pack of 10	$28.00 incl. p&p

UTILITIES

S-KEY by Edwin Paay $15.95 plus 50c. p&p

S-KEY is a complete keyboard driver routine for the TRS-80 and becomes part of the Level II basic interpreter. With S-KEY loaded the user will have many new features not available with the standard machine.

S-KEY features:

* S-KEY provides an auto-repeat for all the keys on the keyboard. If any key is held down longer than about half a second, the key will repeat until it is released.
* Graphic symbols can be typed direct from the keyboard, this includes all 64 graphic symbols available from the TRS-80/SYSTEM 80.
* S-KEY allows text, BASIC commands and/or graphics to be defined to shifted keys. This makes programming much easier as whole commands and statements can be recalled by typing shift and a letter key.
* Because S-KEY allows graphics to be typed directly from the keyboard, animation and fast graphics are easily implemented by typing the appropriate graphics symbols directly into PRINT statements.
* S-KEY allows the user to LIST a program with PRINT statements containing graphics, properly. S-KEY does this by intercepting the LIST routine when necessary.
* S-KEY allows the user to list an updated list of the shift key entries to the video display or line printer.
* S-KEY can be disabled and enabled when required. This allows other routines which take control of the keyboard to run with S-KEY as well.

Each cassette has TRS-80, DISK and SYSTEM 80 versions and comes with comprehensive documentation.

BMON by Edwin Paay $19.95 plus 50c. p&p
THE ULTIMATE HIGH MEMORY BASIC MONITOR L2/16-48K

Our own personnel refuse to write BASIC without first loading this amazing machine language utility program into high memory! BMON Renumbers; Displays BASIC programs on the screen while they are still loading; tells you the memory locations of the program just loaded; lets you stop a load part-way through; merges two programs, with automatic renumbering of the second so as to prevent any clashes of line numbers; recovers your program even though you did type NEW: makes one program invisible while you work on a second (saves hours of cassette time!); lists all the variables used in the program; makes SYSTEM tapes; lets you Edit memory directly . . . the list goes on and on. Cassette comes with 16K, 32K and 48K versions, ready to load. Can anyone afford NOT to have BMON?

EDUCATIONAL

RPN CALCULATOR (L2/16K & 32K)
$14.95 $ 50c. p&p

Give your computer the power of a $650 reverse polish notation calculator with 45 functions and selectable accuracy of 8 or 16 digits. The main stack and registers are continuously displayed whilst the menu is always instantly accessible without disturbing any calculations or register values. The cassette comes with both the 16K and 32K versions, the latter giving you the additional power of a programmable calculator. Comes with a very comprehensive 15 page manual, which includes instructions to load and modify the 32K programmable version to run in 16K. Whether for business or pleasure, this package will prove invaluable, and turn you '80 into a very powerful instrument.

GAMES

MICROPOLY (L2/16K) $8.95 + 60c p&p

Now you can play Monopoly on your micro. The old favourite board game has moved into the electronic era. This computer version displays the board on the screen, obeys all the rules and, best of all, the banker does not make mistakes with your change!

CONCENTRATION (L2/16K) $8.95 + 60c p&p

Another application of supergraphics. There are 28 "cards" displayed on the screen, face down. Players take it in turn to turn them over with the object of finding matching pairs. There are 40 different patterns which are chosen at random, so the game is full of endless variety. This is of particular value in helping young children to learn the art of concentrating and, at the same time, to introduce them to the computer.

METEOR AND TORPEDO ALLEY (L2/16K)
$10.95 + 60c p&p

Those who frequent games arcades will recognize these two electronic games. In METEOR you must destroy the enemy space ships before they see you. In its most difficult mode, the odds are a thumping 238 to 1 against you being successful. In torpedo alley you must sink the enemy ships without hitting your own supply ship. Both games include sound effects and are remarkably accurate reproductions of the arcade games.

AUSTRALIAN SOFTWARE (Cont.)

GAMES

SHEEPDOG (L2/16K) **$8.95 + 60c p&p**
Ever wondered how a sheepdog manages to drive all those awkward sheep into a pen? Well, here is your chance to find out just how difficult it is and have a lot of fun at the same time. You control the sheepdog, the computer controls the sheep! As if that isn't enough, look out for the dingoes lurking in the bush!

U BOAT **$8.95 + 60c p&p**
Real time simulation at its best! Comes with working sonar-screen and periscope, a full rack of torpedoes, plenty of targets, working fuel and battery meters, helpful Mothership for high-seas reprovisioning and even has emergency radio for that terrible moment when the depth charges put your crew at risk. Requires Level II/16K.

SPACE INVADERS WITH SOUND **$8.95 + 60c p&p**
Much improved version of this arcade favourite with redesigned laser and cannon blasts, high-speed cannon, 50 roving drone targets, 10 motherships and heaps of fun for all. Level II with 4K and 16K versions on this cassette.

GOLF (L2/16K) **$8.95 + 60c p&p**
Pit your skills of mini-golf against the computer. Choose the level of difficulty, the number of holes and whether you want to play straight mini golf or crazy golf. Complete with hazards, water traps, bunkers and trees. Great fun for kids of all ages.

DOMINOES(L2/16K) **$8.95 + 60c p&p**
Pit your skill at dominoes against the computer, which provides a tireless opponent. Another application of supergraphics from the stable of Charlie Bartlett. Dominoes are shown approximately life size in full detail (except for colour!). The monitor screen is a window which you can move from one end of the string of dominoes to the other. Best of all, you don't lose any pieces between games!

KID'S STUFF (formerly MMM-1) **$8.95 + 60c p&p**
Three games on one cassette from that master of TRS-80 graphics, Charlie Bartlett. Includes INDY 500, an exciting road race that gets faster and faster the longer you play, SUBHUNT in which your warship blows up unfortunate little submarines all over the place, and KNIEVEL (as in motorcycle, ramp and buses).

OTHER PROGRAMS

INFINITE BASIC BY RACET (32K/1 DISK)
$49.95 + 50c. p&p
Full matrix functions – 30 BASIC commands; 50 more STRING functions as BASIC commands.

GSF/L2/48K **$24.95 + 50c. p&p**
18 machine language routines including RACET sorts.

BUSINESS ADDRESS AND INFORMATION SYSTEM (48K/DISK) **$24.95 + 50c. p&p**
Allows you to store addresses and information about businesses, edit them and print them out.

HISPED (L216, 32 or 48K) $29.95
This machine language program allows you to SAVE and LOAD programs and data to tape at speeds up to 2000 band (4 times normal) using a standard cassette recorder. A switch must be installed to remove the XRX III loading board, if fitted.

LOWER CASE FOR YOUR TRS-80/SYSTEM 80
Kit only $49.00 plus $2.00 p&p

Give your TRS-80 or SYSTEM 80 a lower case display with proper descenders and a block cursor (similar to the TRS-80 Model III). Also includes symbols for the four suits of cards. Includes full fitting instructions, all necessary components and a special machine language driver program to enable lower case in BASIC. The modification is similar to the Tandy model and does not work with Electric Pencil without further modifications.
These kits require disassembly of your computer and some soldering. They should only be installed by someone who has experience in soldering integrated circuits, using a low power, properly earthed soldering iron. If you do not have the necessary experience/equipment, we will install the modification for you for $20 plus freight in both directions. Make sure you arrange the installation with us first, before despatching your computer, so that we can assure you of a rapid turn-around. We are also arranging to have installers in each State. See elsewhere in this issue for their names and addresses.

PRICES

Cat No.
HD-020 Lower case mod kit for TRS-80
$49.00 plus $2.00 p&p
HD-021 Lower case mod kit for SYSTEM-80
$49.00 plus $2.00 p&p

EPSON MX-80 PRINTER
ONLY *$949 Inc. Cable for TRS-80 and p&p
(*Printer only – $940 incl. p&p)

The EPSON MX-80 printer is compact, quiet, has features unheard of only 2-3 years ago in a printer at any price and, above all, is ultra-reliable. All available print modes may be selected under software control. Features include:
- high quality 9x9 dot-matrix character formation
- 3 character densities
  - . 80 characters per line at 10 chars/inch
  - .132 characters per line at 16.5 chars/inch
  - . 40 characters per line at 5 chars/inch
- 2 line spacings
  - . 6 lines per inch 8 lines per inch
- 80 characters per second print speed
- bi-directional printing
- logical seeking of shortest path for printing
- lower case with descenders
- TRS-80 graphics characters built in
- standard Centronics printer port

The bi-directional printing coupled with the logical seeking of the shortest print path (which means that the print head will commence printing the next line from the end which requires the least travel, thereby minimising unutilised time) gives this printer a much higher throughput rate than many other printers quoting print speeds of 120 c.p.s. or even higher.

GREEN SCREEN SIMULATOR
$9.50 incl. p&p

The GREEN SCREEN SIMULATOR is made from a deep green perspex, cut to fit your monitor. It improves contrast and is much more restful to the eyes than the normal grey and white image.
All editorial staff of MICRO-80 are now using GREEN SCREEN SIMULATORS on their own monitors.
Please make sure to specify whether you have an old (squarish) or new (rounded) style monitor when ordering. Not available for Dick Smith monitors.

```
250 DATA 26,24,24,24,24,24,24,24,24,24,24
260 DATA 128, 181,128,128,128,128,128,170,144
270 REM TROLL
280 DATA 128,128,128,128,128,128,128,128
290 DATA 26,24,24,24,24,24,24,24,24
300 DATA 176,176,176,187,183,176,176,176
310 DATA 26,24,24,24,24,24,24,24,24
320 DATA 128,128,170,191,191,149,128,128
330 DATA 26,24,24,24,24,24,24
340 DATA 186,149,170,181
350 REM LITTLE GOAT
360 FOR I=1 TO 22
370 READ L
380 L$=L$+CHR$(L)
390 NEXT I
400 REM MIDDLE-SIZED GOAT
410 FOR I=1 TO 39
420 READ M
430 M$=M$+CHR$(M)
440 NEXT I
450 REM LARGE GOAT
460 FOR I=1 TO 51
470 READ B
480 B$=B$+CHR$(B)
490 NEXT I
500 REM TROLL
510 FOR I=1 TO 53
520 READ T
530 T$=T$+CHR$(T)
540 NEXT I
550 REM INTRODUCE CHARACTERS
560 CLS
570 PRINT"THIS IS THE LITTLE BILLY GOAT GRUFF"
580 PRINT"HE LIKES TO JUMP
590 PRINT@452,L$
600 GOSUB 1930
610 CLS
620 PRINT"THIS IS THE MIDDLE-SIZED BILLY GOAT GRUFF"
630 PRINT"HE LIKES TO HAVE FUN"
640 PRINT@388,M$
650 GOSUB 1930
660 CLS
670 PRINT"THIS IS THE BIG BILLY GOAT GRUFF"
680 PRINT"HE LIKES TO EAT GRASS"
690 PRINT@388,B$
700 GOSUB 1930
710 CLS
720 PRINT"THIS IS THE TROLL"
730 PRINT"HE LIVES UNDER A BRIDGE"
740 PRINT@611,T$
750 GOSUB 1930
760 PRINT@128,"HERE IS THE BRIDGE"
770 GOSUB 1850 :GOSUB 1930
780 REM START STORY
790 PRINT@0,C3$
800 PRINT@0,"THE GOATS ARE HUNGRY":PRINT"THEY WANT TO GO OVER TH
E BRIDGE FOR SOME GRASS"
810 GOSUB 1940
820 PRINT@0,C2$
830 PRINT@0,"HERE COMES LITTLE BILLY GOAT GRUFF"
840 GOSUB 1930
850 GOSUB 1960
860 FOR N=384 TO 410:PRINT@N,L$;:FOR Z=1 TO 20:NEXT Z,N
870 PRINT@0,C2$
880 GOSUB 1930
890 GOSUB 1750
900 GOSUB 1820
910 GOSUB 2000
920 GOSUB 1930
930 PRINT@104,C2$
940 GOSUB 2040
950 PRINT@64,C3$;
960 GOSUB 2090
970 GOSUB 1930
980 GOSUB 2130
990 GOSUB 1820
1000 GOSUB1970
1010 FOR N=411 TO 440:PRINT@N,L$;:FOR Z=1 TO 20:NEXTZ,N
```

```
1020 PRINT@0,C$
1030 PRINT@384,C2$;
1040 PRINT@0,"HERE COMES MIDDLE SIZED BILLY GOAT GRUFF"
1050 GOSUB 1930
1060 GOSUB 1960
1070 FOR N=321 TO 346:PRINT@N,M$;:FOR Z=1 TO 20:NEXTZ,N
1080 GOSUB 1930
1090 PRINT@0,C$
1100 GOSUB 1930
1110 GOSUB 1750
1120 GOSUB 1820
1130 GOSUB 2000
1140 GOSUB 1930
1150 PRINT@104,C2$
1160 GOSUB 2040
1170 PRINT@64,C3$;
1180 GOSUB 2090
1190 GOSUB 1930
1200 GOSUB 2130
1210 GOSUB 1820
1220 GOSUB 1970
1230 FOR N=347 TO 376:PRINT@N,M$;:FOR Z=1 TO 20:NEXTZ,N
1240 PRINT@320,C3$;
1250 PRINT@0,"HERE COMES BIG BILLY GOAT GRUFF"
1260 GOSUB 1930
1270 GOSUB 1970
1280 FOR N=321 TO 340:PRINT@N,B$;:FOR Z=1 TO 10:NEXTZ,N
1290 GOSUB 1930
1300 PRINT@0,C$
1310 GOSUB 1930
1320 GOSUB 1750
1330 GOSUB 1820
1340 GOSUB 2000
1350 GOSUB 1930
1360 PRINT@0,C3$
1370 PRINT@71,"GET OUT OF MY WAY"
1380 GOSUB 1930
1390 PRINT@135,"I'M GOING TO PASS"
1400 GOSUB 1930
1410 PRINT@64,C2$
1420 FOR N=340 TO 345:PRINT@N,B$;:FOR Z=1 TO 20:NEXTZ,N
1430 FOR M=1 TO 4:PRINT@291-M*64,T$;:PRINT@549-M*64,TC$;:NEXTM
1440 FOR M=0 TO 9:PRINT@35+64*M,T$;:NEXTM
1450 GOSUB 1820
1460 FOR M=10 TO 11:PRINT@35+64*M,T$;:NEXTM
1470 TC$=STRING$(8,128):TM$=STRING$(8,182)
1480 PRINT@675+64,TC$;
1490 PRINT@675+128,TC$;
1500 PRINT@675+192,TM$;
1510 PRINT@675+256,TM$;
1520 GOSUB 1970
1530 FOR N=345 TO 373:PRINT@N,B$;:FOR Z=1 TO 20:NEXTZ,N
1540 PRINT@320,C3$;
1550 PRINT@0,C$
1560 GOSUB 1930
1570 CLS
1580 PRINT"HERE IS THE OTHER SIDE OF THE STREAM"
1590 GOSUB 1930
1600 PRINT@0,"HERE IS THE GRASS ON THE OTHER SIDE OF THE STREAM"
1610 FOR N=1 TO 100
1620 R=RND(831)+192:PRINT@R,G$;
1630 NEXT N
1640 GOSUB 1930
1650 PRINT@0,"HERE ARE THE GOATS ON THE OTHER SIDE OF THE STREAM
"
1660 GOSUB 1930
1670 PRINT@395,L$;:PRINT@540,M$;:PRINT@745,B$;
1680 GOSUB1930
1690 PRINT@64,"THEY HAVE LOTS OF GRASS":GOSUB 1930
1700 PRINT@128,"THEY ARE HAPPY"
1710 GOSUB 1930
1720 CLS
1730 PRINTCHR$(23):PRINT@404,"THE END"
1740 GOTO 1740
1750 REM TROLL UP
1760 PRINT@0,"UP JUMPS THE TROLL"
1770 GOSUB 1930
1780 PRINT@0,C$
1790 FOR M= 1 TO 5
```

```
1800 PRINT@611-M*64,T$;:PRINT@869-M*64,TC$;
1810 NEXT M:RETURN
1820 REM REPAIR BRIDGE
1830 FOR X=70 TO 85:SET(X,24):NEXT X:RETURN
1840  END
1850 REM DRAW BRIDGE
1860 FOR X=0 TO 127
1870 SET(X,24)
1880 NEXT X
1890 FOR Y=25 TO 41
1900 SET(47,Y):SET(53,Y):SET(95,Y):SET(101,Y)
1910 NEXT Y
1920 RETURN
1930 REM SPACE-BAR TO PROCEED SEQUENCE
1940 P$=INKEY$:IF P$="" THEN 1940
1950 RETURN
1960 PRINT@0,C$
1970 PRINT@0,"TRIP TRAP, TRIP TRAP, TRIP TRAP"
1980 RETURN
1990 END
2000 REM STOP DIALOGUE
2010 PRINT@104,"STOP"
2020 PRINT@168,"I WANT TO EAT YOU UP"
2030 RETURN
2040 REM EAT BROTHER
2050 PRINT@71,"NO, DON'T EAT ME"
2060 GOSUB 1930
2070 PRINT@135,"EAT MY BROTHER":GOSUB 1930
2080 PRINT@199,"HE IS FATTER THAN I AM":GOSUB1930 :RETURN
2090 REM CROSS BRIDGE SEQUENCE
2100 PRINT@168,"YOU MAY CROSS THE BRIDGE";:GOSUB 1930
2110 PRINT@128,C2$;
2120 RETURN
2130 REM TROLL DOWN
2140 FOR M=1 TO 5
2150 PRINT@291+M*64,T$;
2160 NEXT M
2170 RETURN
```

***** MOVIE L/4K ml. by M. White *****

A machine code routine to allow two new commands in BASIC to provide easy animation of games displays. The idea for these commands came from the game "Super-Sizzler" published in MICRO-80 and, in fact, these commands will drive the assembled car off the screen quite realistically in a modified "Super-Sizzler" game.

The commands are unused Disk BASIC commands and are suitable for use with Level II BASIC only. They operate by scrolling a range of lines on the screen, either left or right and with or without wraparound. A single line will scroll much faster than a full screen. Scroll speed is controlled in BASIC using a timing loop.

The two commands are:

LSET (Arg1, Arg2, Arg3)
RSET (Arg1, Arg2, Arg3)

LSET moves a block of lines one field left.
RSET moves a block of lines one field right.
Arg1 is the top line of the block move in the range Ø to 15 and may be a constant or variable of any type except string.
Arg2 is the bottom line of the block move and may be a constant or variable of any type except string.
Arg 3. If the value is Ø the scrolling will wrap around - if any other value it will run off the screen. It similarly can be a constant or variable.

Out of range arguments will return values set to limit of legal range instead of error.

Typical uses in games would be to set up rows of moving targets such as ducks or spaceships. Different rows of targets can be moving in opposite directions and/or at different speeds. One word of warning, any missile or bullet moving up through a moving zone will itself be moved sideways by these routines and this will need to be compensated for.

The source code listing is well documented with remarks. Some of the more significant routines are:

- EVAL which is called to evaluate the argument to the command. It sets the HL register to point to the start of each argument and calls GET to return the value as a screen address.

- Routine 'GET' tests whether the argument is a constant or a variable and converts it to an integer constant. This is tested for range error and any out of range value is reset to the nearest legal value in the range Ø to 15. It is then multiplied by 4ØH and added to 3CØØH to return a screen address.

  The main routine then executes a 'LDDR' or 'LDIR' instruction to move a line on the screen. It then calls routine 'UPDATE' to check if that is the last line to be moved and to wrap around the line if required.

If you have an Editor/Assembler, you may type in the source code listing (columns 3, 4, 5 and 6). You may then Assemble the code and punch an object tape.

If you do not have an Editor/Assembler, you may use a monitor program such as BMON. (TBUG is not satisfactory as it loads into the same area in memory as MOVIE). (BMON was published in MICRO-80 Issues 3, 4 and 5 or is available on cassette for $19.95). To enter the program, use the Edit Memory function, starting at 42E9 Hex. The hex listing is contained in columns 1 and 2 of the listing. Change the value stored at 42E9 to 21, at 42EA to 99, at 42EB to 44, at 42EC to AF, at 42ED to 77 and so on until you reach 4498 which should be changed to C9. Break out of the Edit mode and Copy memory to tape using the following parameters:

START	END	ENTRY	NAME
42E9	4498	42E9	MOVIE

To load from tape, type in SYSTEM, answer *? with MOVIE, press ENTER/NEW LINE. Answer the second *? with /, press ENTER/NEW LINE.

```
42E9              00010          ORG     42E9H
                  00020 ;        RESET BASIC POINTERS PAST M/C ROUTINE.
42E9 219944       00030 INIT     LD      HL,FINISH+1
42EC AF           00040          XOR     A
42ED 77           00050          LD      (HL),A
42EE 23           00060          INC     HL
42EF 77           00070          LD      (HL),A
42F0 22A440       00080          LD      (40A4H),HL
42F3 23           00090          INC     HL
42F4 77           00100          LD      (HL),A
42F5 23           00110          INC     HL
42F6 22F940       00120          LD      (40F9H),HL
42F9 22FB40       00130          LD      (40FBH),HL
42FC 22FD40       00140          LD      (40FDH),HL
                  00150 ;        SET JUMP ADRESSES FOR 'LSET' AND 'RSET' COMMANDS.
42FF 211744       00160          LD      HL,RSET
4302 229B41       00170          LD      (419BH),HL
4305 21F943       00180          LD      HL,LSET
4308 229841       00190          LD      (4198H),HL
430B 211443       00200          LD      HL,TITLE
430E CDA728       00210          CALL    28A7H
4311 C3CC06       00220          JP      6CCH
4314 4D           00230 TITLE    DEFM    'MOVIE:- A BLOCK GRAPHICS MOVE ROUTINE USING
 COMMANDS                   LSET(A1,A2,A3) AND RSET(A1,A2,A3).                  '
4387 20           00240          DEFM    '                COPYRIGHT  20/4/81  MICHAEL WH
ITE                         19 LAWLEY CRES. SOUTH HOBART TAS. 7000."'
0008              00250 BUFF     DEFS    8H
0002              00260 POINT    DEFS    2H
43F9 CD3444       00270 LSET     CALL    EVAL     ;EVALUATE ARGUMENTS
43FC 013F00       00280 LOOPA    LD      BC,3FH
43FF 2AEF43       00290          LD      HL,(BUFF)
4402 ED5BEF43     00300          LD      DE,(BUFF)
4406 23           00310          INC     HL       ;SET REGISTERS FOR 'LDIR'
4407 1A           00320          LD      A,(DE)   ;SAVE FIRST FIELD
4408 32F543       00330          LD      (BUFF+6),A
440B EDB0         00340          LDIR             ;BLOCK MOVE
440D CD7B44       00350          CALL    UPDATE
4410 38EA         00360          JR      C,LOOPA
4412 2AF743       00370          LD      HL,(POINT)       ;RETURN PROGRAM POINTER
4415 23           00380          INC     HL
4416 C9           00390          RET                      ;RETURN TO BASIC.
4417 CD3444       00400 RSET     CALL    EVAL     ;EVALUATE ARGUMENTS
441A 013F00       00410 LOOPB    LD      BC,3FH
441D 2AEF43       00420          LD      HL,(BUFF)
4420 09           00430          ADD     HL,BC
4421 E5           00440          PUSH    HL
4422 D1           00450          POP     DE
4423 2B           00460          DEC     HL       ;SET REGISTERS FOR 'LDDR'
4424 1A           00470          LD      A,(DE)
4425 32F543       00480          LD      (BUFF+6),A
4428 EDB8         00490          LDDR                     ;BLOCK MOVE
442A CD7B44       00500          CALL    UPDATE
442D 38EB         00510          JR      C,LOOPB
442F 2AF743       00520          LD      HL,(POINT)       ;RETURN PROGRAM POINTER.
```

```
4432 23       00530          INC    HL
4433 C9       00540          RET
              00550 ;        EVAL  EVALUATE ARGUMENTS OF COMMAND.
4434 CD4D44   00560 EVAL     CALL   GET
4437 22EF43   00570          LD     (BUFF),HL
443A 2AF743   00580          LD     HL,(POINT)
443D CD4D44   00590          CALL   GET
4440 22F143   00600          LD     (BUFF+2),HL
4443 2AF743   00610          LD     HL,(POINT)
4446 CD4D44   00620          CALL   GET
4449 22F343   00630          LD     (BUFF+4),HL
444C C9       00640          RET
              00650 ;  GET   RETURN INTEGER VALUE OF ARGUMENT. RANGE 0 - 15
444D 23       00660 GET      INC    HL         ;MOVE HL TO FUNC. ARG.
444E 7E       00670          LD     A,(HL)
444F FE3C     00680          CP     3CH        ;TEST IF VARIABLE OR CONST.
4451 3005     00690          JR     NC,VAR
4453 CD650E   00700          CALL   0E65H      ;RETURN INTEGER IN 'ACC'
4456 1803     00710          JR     NEXT1
4458 CD4025   00720 VAR      CALL   2540H      ;RET. VALUE OF VAR. IN 'ACC'
445B 22F743   00730 NEXT1    LD     (POINT),HL
445E CD7F0A   00740          CALL   0A7FH      ;CONVERT TO INTEGER
              00750 ;   TEST FOR RANGE ERRORS. IF ERROR RESET TO RANGE LIMIT.
4461 7D       00760          LD     A,L
4462 FE00     00770          CP     0
4464 3002     00780          JR     NC,ZNEX1
4466 2E00     00790          LD     L,0
4468 FE0F     00800 ZNEX1    CP     0FH
446A 3802     00810          JR     C,ZNEX2
446C 2E0F     00820          LD     L,0FH
446E 2600     00830 ZNEX2    LD     H,0
4470 29       00840          ADD    HL,HL
4471 29       00850          ADD    HL,HL
4472 29       00860          ADD    HL,HL
4473 29       00870          ADD    HL,HL
4474 29       00880          ADD    HL,HL
4475 29       00890          ADD    HL,HL
4476 11003C   00900          LD     DE,3C00H
4479 19       00910          ADD    HL,DE      ;GET SCREEN ADRESS
447A C9       00920          RET
447B 3AF343   00930 UPDATE   LD     A,(BUFF+4)          ;WRAP AROUND VALUE
447E FE00     00940          CP     0
4480 2004     00950          JR     NZ,SKIP             ;SKIP IF NOT '0'
4482 3AF543   00960          LD     A,(BUFF+6)          ;WRAP GRAPHICS AROUND
4485 12       00970          LD     (DE),A
4486 ED5BEF43 00980 SKIP     LD     DE,(BUFF)
448A 214000   00990          LD     HL,40H
448D 19       01000          ADD    HL,DE
448E 22EF43   01010          LD     (BUFF),HL
4491 ED5BF143 01020          LD     DE,(BUFF+2)
4495 CD901C   01030          CALL   1C90H      ;TEST END OF LOOP
4498 C9       01040 FINISH   RET
42E9          01050          END    INIT
00000 TOTAL ERRORS
```

***** BASIC ARRAY SAVER/LOADER L2/4K   by K. Shillito *****

This program was written in response to a request way back in MICRO-80 Issue 8. The BASIC listing is heavily REMarked so little explanation is required here. A machine language routine is POKEd into memory by the BASIC program. This machine language routine was hand assembled. Its listing is unusual in that the code in column 2 is shown in decimal rather than the more common Hexadecimal. However, this does make the BASIC program easier to follow. You will note that the DATA in line 117Ø corresponds to the object code in the machine language listing.

If the array being operated on is not a string array, the BASIC program POKEs two JR commands to skip the "string only" commands. Note also that LD HL,Ø and LD DE,Ø near the beginning have the zeroes replaced by AD% and BY% when POKEd by BASIC. In order to avoid excessive POKEing, some LOAD or SAVE column commands needed in only one column but harmless in the other, are shared;

Although the BASIC code is a little convoluted, if you follow the GOSUB's carefully you will see that all the BASIC code does is the abovementioned POKEing, prompt CASSETTE READY and USR. The reason the code is written in this knotty way is to make it as compact as possible while allowing for an orderly pattern of entry addresses.

Note that, since BASIC uses VARPTR, all items must be defined prior to calling (see the REM statements in the BASIC listing) or else BASIC could relocate them when defining new variables, with disastrous results.

MACHINE CODE FOR ARRAY SAVE AND LOAD

Code	Addr.	LOAD Label	LOAD Opcode	LOAD Operands	SAVE (String only) Opcode	SAVE (String only) Operands	Comments
			ORG	573BH			; UNUSED IN 'RESERVED' MEM
62,Ø	16466	START	LD	A,Ø			; DRIVE #-1
2Ø5,18,2	8		CALL	212H			; SWITCH ON MOTOR
2Ø5,15Ø,2	71		CALL	296H	CALL	287H	; READ/WRITE LEADER
33,Ø,Ø	4		LD	HL,Ø			; AD% POKED BY BASIC
17,Ø,Ø	7		LD	DC,Ø			; BY% POKED BY BASIC
27	80	NEXT	DEC	DE			; DEC BYTE/STRING COUNTER
2Ø3,122	1		BIT	7,D			; ANY MORE BYTES/STRINGS?
32,37	3		JR	NZ,OFF			; STRING ROUTINE FOLLOWS:
7Ø	5		LD	B,(HL)			; STRING LEN IS DJNZ FACTOR
35	6		INC	HL			; (HL)=LSB STR ADDR
126	7		LD	A,(HL)			; A = LSB STR ADDR
35	8		INC	HL			; (HL) = MSB STR ADDR
229	9		PUSH	HL			; SAVE STRING ADDR
1Ø2	90		LD	H,(HL)			; H =MSB STR ADDR
111	1		LD	L,A			; (HL) = STRING ADDR
175	2		XOR	A			; A = Ø
184	3		CP	B			; CP Ø, LEN
4Ø,22	4		JR	Z,NULL			; NULL STRING?
58,64,56	6	IO	LD	A,(384ØH)			; BREAK KEY ADDRESS
254,4	9		CP	4			; BREAK PRESSED?
4Ø,19	501		JR	Z,OFF			; STOP IF SO
197	3		PUSH	BC			; SAVE B (STRING BYTE CTR)
1,255,Ø	4		LD	BC,ØFFH			; DELAY FACTOR
2Ø5,44,2	7		CALL	22CH	CALL	6ØH	; BLINK/DELAY
2Ø5,53,2	10		CALL	235H	LD	A,(HL)	; READ BYTE/RETRIEVE BYTE
119	3		LD	(HL),A	CALL	264H	; SAVE BYTE/WRITE BYTE
193	4		POP	BC			; RESTORE B (STRING BYTE CTR)
35	5		INC	HL			; BUMP SAVE/RETRIEVE ADDR
16,234	6		DJNZ	IO			; STRING FINISHED?
225	8	NULL	POP	HL			; RESTORE STRING ADDR
35	9		INC	HL			; NEXT STRING ADDR
24,214	2Ø		JR	NEXT			; NEXT BYTE/STRING
2Ø5,248,1	2	OFF	CALL	IF8H			; SWITCH OFF MOTOR
2Ø1	5		RET				; BACK TO BASIC
82,64	6		DEFW	573BH			; USR POINTERS
			END	START			

; HAND ASSEMBLE FOR BASIC USR
; ROUTINE, DEC 198Ø
;

```
10 '***                                                      ***
20 '*** BASIC ARRAY SAVE/LOAD ROUTINE (LEVEL 2,4K/16K)       ***
30 '*** ===========================                          ***
40 '
50 'KEN SHILLITO          SUBMITTED TO MICRO-80 MAGAZINE JAN 1980
60 '10 MILTON ST.,        (IN RESPONSE TO A REQUEST BY CHRIS
70 'CHATSWOOD 2067        GRIFFIN IN ISSUE #8)
80 '
90 '********** FUNCTION OF PROGRAM **********
100 'THE SUBROUTINE SAVES AND LOADS FILES CONTAINING THE ENTIRE
110 'CONTENTS OF AN ARRAY TO CASSETTE.
120 '
130 '********** RULES FOR USE **********
140 '1.AT THE TIME OF CALLING,VARIABLES I% AND J% MUST ALREADY
150 '  HAVE A VALUE. (THE VALUES ARE IMMATERIAL & WILL CHANGE).
160 '2.A VARIABLE BY% MUST BE GIVEN A VALUE EQUAL TO THE NUMBER
170 '  OF BYTES IN NUMERIC ARRAYS,OR THE NO. OF STRINGS. A
180 '  FURTHER VARIABLE AD% MUST BE SET TO THE VARPTR OF THE
190 '  FIRST ELEMENT OF THE ARRAY. (I.E. WITH EACH SUBSCRIPT 0)
200 '3.WHEN LOADING STRINGS,DUMMY STRINGS OF THE CORRECT LEN
210 '  MUST ALREADY EXIST. (IF THE STRING LENGTH IS NOT CONSTNT
220 '  A LIST OF STRING LENGTHS MUST BE SAVED ALSO AS IN THE
230 '  SAMPLE RUN BELOW).
240 '4.ARRAYS LOADED MUST BE OF THE SAME TYPE AS ARRAYS STORED.
250 '5.THERE ARE NO RESTRICTIONS IN THE CHARACTERS THAT MAY
260 '  APPEAR IN STRINGS.
270 '6.ALWAYS SET AD% IMMEDIATELY BEFORE GOSUB (RULES 1 & 6 ARE
280 '  NEEDED SINCE OTHERWISE BASIC MIGHT MOVE THE ARRAY).
290 '7.ENTRY POINTS:
300 '   TO SAVE A NUMERIC ARRAY          GOSUB 1080
310 '   TO LOAD A NUMERIC ARRAY          GOSUB 1090
320 '   TO SAVE A STRING  ARRAY          GOSUB 1100
```

```
330 '   TO LOAD A STRING  ARRAY         GOSUB 1110
340 '
350 '********** FURTHER COMMENTS **********
360 '1.TO USE #-2,CHANGE THE SECOND ITEM IN LINE 1080 TO 1
370 '2.THE FIGURE 255 NEAR THE LHS OF ROW 3 OF LINE 1080 MAY
380 '  BE REDUCED TO INCREASE THE PACKING DENSITY AND SPEED
390 '  DEPENDING ON THE QUALITY OF THE TAPE INTERFACE. 255
400 '  GAVE NO ERRORS AT ALL IN A SERIES OF TEST RUNS TOTALLING
410 '  128K BYTES WITH WELL USED MEDIUM QUALITY TAPE.
420 '3.WHEN CASSETTE READY? APPEARS,PRESS ANY KEY EXCEPT BREAK
430 '  TO COMMENCE LOADING/SAVING. CHANGE LINE 1120 TO A REM IF
440 '  THIS STEP IS NOT REQUIRED
450 '4.IF THE ASTERISKS STOP BLINKING AT A BLURRING SPEED IN A
460 '  LOAD,SOMETHING HAS GONE WRONG. TO INTERRUPT A SAVE OR
470 '  LOAD,PRESS BREAK.
480 '5.THE SUBROUTINE MAY ALSO BE USED TO SAVE A BLOCK OF MEMRY
490 '  OR LOAD ONE UNDER PROGRAM CONTROL. BY ASTUTE USE OF
500 '  VARPTR AND CHOICE OF LENGTH OF DUMMY STRINGS,ETC. IT IS
510 '  ALSO POSSIBLE TO REARRANGE STRINGS ON RELOADING,LOAD OR
520 '  SAVE PART ONLY OF AN ARRAY,ETC.
530 '6.TO CALCULATE BY%,MULTIPLY 1 MORE THAN EACH DIMENSION BY
540 '  1 FOR STRINGS,2FOR INTEGERS,4FOR SINGLE PRECISIONS,
550 '  AND 8 FOR DOUBLE PRECISIONS. E.G.:
560 '   DIM A$(10):BY%=(10+1)*1
570 '   DIM A#(10,5):BY%=(10+1)*(5+1)*8
580 '7.THE MACHINE CODE IS POKED INTO THE USR PORTION OF MEMORY
590 '  AT 4047H-408FH. HENCE,NO MEMORY RESERVATION IS NEEDED.
600 '8.THE SUBROUTINE USES DATA. WHEN LOADING,IT DOES NOT READ
610 '  THE LAST 6 ITEMS. THE ROUTINE BEGINS WITH A RESTORE.
620 '9.IT IS GOOD PRACTISE TO USE A DUMMY ITEM OF KNOWN VALUE
630 '  AS THE LAST ARRAY ELEMENT FOR CHECKING; A HASH TOTAL CAN
640 '  ALSO BE USED. THE FILES CREATED DO NOT USE FILE NAMES OR
650 '  CHECK DIGITS,AND ALL DATA IS STORED IN 1 LARGE BLOCK.
660 '
670 '***
680 '*** SAMPLE PROGRAM TO ILLUSTRATE HOW TO SAVE AND LOAD NUM-
690 '*** ERIC & STRING ARRAYS USING THE BASIC ARRAY SAVER/LOADR
700 '***
710 '
720 '*** SET UP ARRAY TO BE SAVED ***
730 CLEAR2000:DEFSTR Y-Z:DEFINT A-X:DIM Y(20),Z(20),I(20),J(20)
740 J=0
750 FOR I=0 TO 20:LET Z(I)=STRING$(RND(11)-1,CHR$(64+RND(26)))+
   " ":NEXT
760 '
770 '*** PRINT PRELIMINARY MESSAGE ***
780 CLS:PRINT"*** BASIC SAVER/LOADER DEMONSTRATION PROGRAM ***"
790 PRINT:PRINT"I WILL PRINT CASSETTE READY TWICE, SINCE I ";
800 PRINT"FIRST SAVE THE STRINGLENGTHS, THEN THE STRING ARRAY";
810 PRINT". THERE IS NO NEED TO LEAVE BLANK SPACE ";
820 PRINT"BETWEEN THE FILES - JUST PRESS ANY KEY WHEN CASSETT";
830 PRINT"E READYAPPEARS":PRINT
840 '
850 '*** NOW,WE SAVE THE ARRAYS ***
860 FOR I=0 TO 20:PRINT Z(I);:LET I(I)=LEN(Z(I)):NEXT:PRINT
870 BY%=42:AD%=VARPTR(I(0)):GOSUB1080:PRINT@640,STRING$(30," ")
880 BY%=21:AD%=VARPTR(Z(0)):GOSUB1100:PRINT@640,STRING$(30," ")
890 '
900 '*** MESSAGES FOR RELOAD ***
905 PRINT
910 PRINT"NOW, PLEASE REWIND THE CASSETTE READY FOR RELOADING";
920 PRINT". IF THE":PRINT"ASTERISKS STOP BLINKING AT A BLURRI";
930 PRINT"NG RATE, SOMETHING IS AMISS, WHENCE HIT BREAK & PRE";
940 PRINT"SS GOTO 900 FOR ANOTHER TRY
950 '
960 '*** NOW,WE RELOAD THE ARRAY ***
970 BY%=42:AD%=VARPTR(J(0)):GOSUB1090:PRINT@640,STRING$(30," ")
975 FOR I=0 TO 20:Y(I)=STRING$(J(I),"#"):NEXT:'DUMMY FILL Y
980 BY%=21:AD%=VARPTR(Y(0)):GOSUB1110:PRINT@640,STRING$(30," ")
990 '
1000 '*** CHECK THAT EVERYTHING IS OK & REPORT ***
1005 PRINT@768,;
1010 FOR I=0 TO 20:IF Y(I)=Z(I) PRINT Y(I);:NEXT ELSEPRINT:PRINT
     "READ ERROR":PRINT:GOTO910
1020 PRINT:PRINT:PRINT"*** SAVED & RELOADED ARRAYS MATCH":END
1030 '
1040 '***                                                   ***
1050 '*** BASIC ARRAY SAVER/LOADER,BY KEN SHILLITO           ***
1060 '***                                                   ***
```

```
1070 '
1080 GOSUB1140:GOSUB1160:GOSUB1150:GOTO1120
1090 GOSUB1140:GOSUB1160:GOTO1120
1100 GOSUB1140:GOSUB1150:GOTO1120
1110 GOSUB1140
1120 IF INKEY$=""PRINT@640,"CASSETTE READY?":GOTO 1120
1130 I%=USR(0):RETURN
1140 POKE16553,255:RESTORE:FORI%=16466TO 16527:READ J%:POKE I%,J
%:NEXT:I%=VARPTR(AD%):POKE 16475,PEEK(I%):POKE 16476,PEEK(I%+1):I
%=VARPTR(BY%):POKE 16478,PEEK(I%):POKE 16479,PEEK(I%+1):RETURN
1150 POKE16472,135:FORI%=16508TO16513:READ J%:POKE I%,J%:NEXT:RE
TURN
1160 POKE16485,24:POKE16486,9:POKE 16516,24:POKE 16517,2:RETURN
1170 DATA 62,0,205,18,2,205,150,2,33,0,0,17,0,0,27,203,122,32,37
,70,35,126,35,229,102,111,175,184,40,22,58,64,56,254,4,40,19,197,
1,255,0,205,44,2,205,53,2,119,193,35,16,234,225,35,24,214,205,248
,1,201,82,64,96,0,126,205,100,2
```

***** LOWER CASE DRIVER FOR THE E.S.F. by K. Shillito *****

The MICRO-80 lower case mod requires a software driver to make it work. The usual version that MICRO-80 supplies is well suited for cassette and disk users, but is incompatible with the ESF for the following reasons:

1. Its re-location method zaps the ESF floating bytes.
2. The idea of storing machine code on the VDU prior to re-location doesn't work for the ESF, since it is liable to poke DONE into it.

I have written the program listed below. If you are an ESF user, adopt the following procedure:

(a) Initialize the ESF.

(b) Type in the program, or (yuk!) CLOAD it if you are a cassette subscriber. The code will of course be in lower case if you type it, but you can EDIT it if you wish.

(c) @SAVE your program on a 5' wafer.

Whenever you want lower case, you need only @LOAD and RUN the program. You can also add NEW (not @NEW!) at the end as the last statement, if you want it to delete itself once it has done its thing. The program prints instructions. If you will usually want a blinking cursor, then incorporate POKE 1641Ø,255 anywhere in the program, to save typing it. Note that ESF operations switch off the blinking.

You will note that this driver, because it makes use of the ESF "firmware", is very compact - it only takes 78 bytes (or, in a sense, none, since it uses memory unavailable to BASIC). The similarly optioned cassette version takes 17Ø bytes. Any program that will RUN without OM Error in BASIC will RUN with this, but that is not so with the cassette version.

But, there is a but! If you have a program that uses DOS link addresses (if you don't know what they are, you don't, so don't worry), or which generates L3 Error, then you must re-locate the program at high memory, e.g. by specifying 3269Ø for MEMORY SIZE, and let I=3269Ø in line 5Ø.

```
10 DATA 42,32,64,218,154,4,58,34,64,183,40,1,119,121,254,32,218,
6,5,254,128,210,166,4,195,125,4
20 DATA 229,33,26,64,126,43,166,40,12,53,32,9,54,128,42,32,64,12
6,238,127,119,205,92,55,79,33,24,64,254,32,32,5,58,128,56,174,119
,126,225,183,121,200,254,65,216,254,91,200,238,32,201
30 DATA 33,0,0,34,30,64,33,0,0,34,22,64,201
40 DEFINT I-J
50 I=16722        :'CHANGE THIS FIGURE TO RELOCATE THE DRIVER
60 FOR J=I TO I+77:READ K:POKE J,K:NEXT
70 FOR J=16455 TO J+12:READ K:POKE J,K:NEXT
80 POKE 16457,I/256:POKE 16456,I-256*PEEK(16457):POKE 16463,(I+2
7)/256:POKE 16462,I+27-256*PEEK(16463):POKE 16409,1
90 POKE 16526,71:POKE 16527,64:I=USR(0)
100 CLS:PRINT "Lower Case Driver (E.S.F. Version) is now Operati
ve":PRINT:PRINT "c/- Ken Shillito, 10 Milton St., Chatswood 2067,
  July, 1981"
110 PRINT "For blinking cursor, POKE 16410,255. To stop blink, p
oke 0":PRINT "Press <SHIFT>0 to lock/unlock shift, and then backs
pace":PRINT "To relocate, note the REM in line 50"
```

***** SOUND EFFECTS REVISITED (OR $TRING$ AND THING$) by Ronald J. Sully *****

Requires 16K or more of memory and is Disk Basic compatible.

SOUND EFFECTS REVISITED is a utility program written in Level II Basic (Disk compatible) specifically to allow you, the user, to create your own data tables, and/or discover data tables already resident in ROM, for sound effects which can then be incorporated in your own Basic programs, simply and efficiently.

The complete program as listed occupies 6113 bytes of RAM. The actual utility (lines 1Ø-19Ø) require only 927 bytes of RAM plus 6Ø bytes of reserved RAM for the M/L routine. The M/L routine is relocatable and the program sets its own MEMORY SIZE to 6Ø bytes below current top of memory then POKEs the M/L routine into that memory area. For example: if you have 16K of memory the MEMORY SIZE will be set to 327Ø7 automatically by the program then the M/L routine will be POKEd into memory locations 327Ø7 to 32754. With 48K of memory the memory locations reserved will be from 65475 to 65522 (-61 to -14). The start address of the M/L routine is then POKEd into memory locations 16526 and 16527 for Level II users or DEFined by the DEFUSR statement if you are using Disk Basic. There's no need to make any amendments to the program; it knows what sort of a system you are using.

The M/L routine is called by the X=USR statement and the sounds created depend on the value inside the brackets of the X=USR(n) statement. That value in the brackets is used by the M/L routine as a pointer. It points to a memory area in ROM or RAM. If the statement is X=USR(66) then the M/L routine will use the values in memory location 66 and 67 for the note and the value in 68 (Ø) as an end byte. If the statement is X=USR(SA(n)) then SA(n) points to the start address of string SS$(n) and the M/L routine uses the ASCII value of the characters in SS$(n) as data values for the notes.

Lines 1Ø to 19Ø of the program comprise the actual utility and include the data for the M/L routine and the logic to determine MEMORY SIZE and the address in RAM of the strings used and must be included in your own program for sound effects. Lines 23Ø upward contain all the logic for you to:

(a) simply and quickly create your own sound effects (stored in strings),

(b) search ROM and RAM for useful sound routines,

(c) play the sounds before deciding they are suitable for your program, and

(d) simply to let you fool around with sound to your heart's content.

To get a better understanding of the program, let's RUN it. (There's no need to set MEMORY SIZE). If you are keying it in, take note that the strings in lines 8Ø to 17Ø can be any length up to 6Ø characters and must contain an even number of bytes and don't put the quotation mark " at the end. The "a", "c", "d" etc. in lines 49Ø to 55Ø are all lower case (shifted) characters. Make sure you CSAVE before RUNning and think again about spending a mere $36 extra for the cassette edition.

When you are satisfied that the program is either keyed in or CLOADed correctly, connect an amplifier, turn it on and let's get underway. RUN.

The first thing you see on the VDU (besides the MEMORY SIZE message) is a list of 4 options and a live prompt waiting for your selection:

OPTION 1. BUILD OR EDIT $TRING

This option allows you to select your own values for both the Duration and Frequency. (See SOUND EFFECTS, MICRO-8Ø Issue 8, July 1980, page 36 - by yours truly). These values will be stored in 1 of 10 strings in lines 8Ø to 17Ø for future use. There are some special function keys designed to make assembly of the strings quicker and easier. These keys are all shifted except for the up and down arrows.

SHIFTED KEY	FUNCTION	MEANING
A	ABORT	Cancel changes, restore string to previous values and return cursor to byte 1. 'Similar to 'A' in Level II Edit mode).
C	COPY	Copy the value of the previous like byte. If the cursor is pointing to a DURation byte then that byte will take on a value the same as the previous Duration byte. Can only be used if byte number is greater than 2.
D	DECREMENT	Decrement the value of the previous like byte by the INC/DEC value selected earlier and store that value in the current byte.

SHIFTED KEY	FUNCTION	MEANING
E	END & EXIT	End all editing, save the changes, check the validity of all entries and return to options. (Similar to 'E' in Level II Edit mode).
I	INCREMENT	Like DECREMENT but value is incremented.
P	PROPORTIONAL	Subtract the value of the previous unlike byte from 256 and store it in the current byte.
R	RANDOM	Select a random number between 1 and 255 and store it in the current byte.
S	SAME	Take the value of the previous unlike byte and store it in the current byte.
DOWN ARROW/ ESCP		Move cursor to next byte. Will continue until key is released.
UP ARROW CNTRL		Move cursor to previous byte. Will continue until key is released.
ENTER/NEW LINE		Move cursor to next byte. (Has other function explained later).

Practice makes perfect so let's select option 1 and give it a go. The first thing we must decide is on which of the 10 strings (Ø-9) we want to work. Seeing it's our first run through, we will select string 9 (the longest). Press the '9' key and you will be asked for the INCREMENT/ DECREMENT value. You can select any value here but it would be pointless to select large numbers. For demonstration purposes, enter the value 2. (If you want Ø you can simply press ENTER/NEW LINE). Having done that we see at the top of the VDU a reminder of the special function keys, the number of the string we are working on and the value we selected for the INCREMENT/DECREMENT. The 6 columns on the screen relate to the byte number (from 1 up to 6Ø) and the value of that byte. (The value can be from 1 to 255 excluding 34). The columns are headed DUR (duration) and FREQ (frequency). Each pair of bytes (1&2, 3&4, 5&6 etc.) represent 1 note. The current value of all the bytes should be 42 which is the ASCII code for the asterisk "*". In other words, SS$(9) = "******** (60 of them).

You couldn't help but notice that the cursor, (a funny looking arrow), is pointing to the first byte. Let's try some of our special function keys. Firstly the arrows; press /ESCP and watch the cursor advance. Hold the key down and you will see it advance pass 6Ø to start at 1 again. Try the /CNTRL similarly. Use these keys to move the cursor to the particular byte you want to edit. Press the ENTER/NEW LINE key. It simply advances the cursor 1 byte. Move the cursor back to 1 and we'll use some of the other special function keys. Remember to hold the SHIFT key down for the alpha special function keys. Press SHIFT R; byte number 1 has taken on a value between 1 and 255 selected at random. (I got 171). Do that several times and you will notice that the cursor will automatically advance to the next byte. If you have finished playing with the R key we will have a go at the A key. Press SHIFT A; Bingo! We are back where we started from. We just aborted and cancelled all our editing so all the bytes now have the value of 42 again. Now type your name. That's right; just type ROBERT ALFRED BLOGGS (if that is your name). Just type normally all unshifted. How about that? Each byte took on the ASCII value of the keys you pressed. It works for all characters with the ASCII codes 32 to 47 and from 58 to 9Ø. Press SHIFT A again.

Now comes the important part, so pay attention! Press "Ø" twice, then "1". The value in byte 1 should read "ØØ1" and the cursor is pointing to byte 2. Press "Ø" then press "1". Now press ENTER/NEW LINE. The value in byte 2 should read "Ø1" and the cursor is pointing to byte 3. Now press "1" then ENTER/NEW LINE. The value in byte 3 should read "1", and each of the bytes 1, 2 and 3 contain the value of 1. The important thing to remember is the byte will take on the value keyed in and the cursor will advance to the next byte whenever:

(a) three digits are keyed in, or

(b) When the ENTER/NEW LINE key is pressed after 1 or 2 digits.

Press SHIFT A to abort.

Make byte 1 equal to 5 and byte 2 equal to 2, OK? Good, because we are going to create our first sequence of notes. The cursor should now be pointing to byte 3. Hold the SHIFT key down and press "C". Byte 3 is now the same as byte 1 because we Copied it. The cursor is now pointing to byte 4. While still holding down the SHIFT key, press "I". Byte 4 is now equal to 4 because we Incremented it by our INC/DEC value of 2. Continue in this manner, SHIFT C,I,C,I,C,I etc. until all the odd numbered bytes (DURation) contain 5 and all the even numbered bytes (FREQuency) are 2 greater than the previous frequency byte. That is to say, we have selected a relative value of 5 for the duration of all our notes and the frequency of the notes will gradually get higher.

Now we will try another special function key. Press SHIFT 'E'. (End & Exit). Well! We ended alright but we didn't exit. We've been presented with a REDO message at the bottom of the screen and the cursor is pointing to byte number 34 which contains the value of 34. (Coincidence)..

52707

The reason is that I have written the program so the value of 34 is invalid for any of the bytes. ASCII 34 is the quotation mark " . It would not matter when playing the notes but would produce an error message if you CSAVEd, CLOADed then RUN. So, to be on the safe side, we will preclude the value of 34 from any of our byte values. Type "33" ENTER/NEW LINE, or "Ø33", or SHIFT 1(!) - all are equal to 33. Now press SHIFT 'E' again. Whalla! (An old Arabic exclamation statement). We are back to the options. We have just completed creating our first sound string and the lesson on OPTION 1.

OPTION 2. PLAY $TRING NOTES

Select Option 2. Now press '9' and hold the key down. How about that? Shades of Buck Rogers! While we are here, we may as well press all the number keys (Ø-9) and hear what the others sound like.

The centre of the screen shows the statement you need to use if you want that sound in your BASIC program - "X" = USR (SA(9)) " - that is, of course, if you have appended your program to lines 1Ø-19Ø of this program and you have an amplifier connected. So far so good. Now press 'R' (unshifted) to get back to the options because there is no more to say about option 2.

Select Option 1 and choose a string to edit. Perhaps number 8. We are now going to use the SHIFT 'P' and SHIFT 'S' keys. Put 4Ø into byte 1. Hold down the SHIFT key and press 'P'. Byte 2 has now taken the value of 256-4Ø = 216. Keep the SHIFT key down and press 'S'. Continue pressing P,S,P,S ... until all the DUR bytes alternate from 4Ø to 216 and the FREQ bytes alternate from 216 to 4Ø. (Byte 54 = 216). SHIFT 'E' will assemble the string and present you with the options again. Option 2 will allow you to play the string you just created.

We have now used all the special function keys available with Option 1 except SHIFT 'D'. Try it for yourself. It works the same as SHIFT 'I' except, of course, the value is Decremented by the INC/DEC value.

One more sound string before we get into the other options. Select Option 1. Select a string (say number 7). Enter 1 for the INC/DEC. Make byte 1 equal to 4 and byte 2 equal to 11. Press SHIFT C,I,C,I ... etc. until all DUR Bytes = 4 and FREQ bytes start at 11 and the value is incremented until byte 48 = 34. SHIFT 'E' will get you a REDO message at byte 48. Fix it!

We have now covered Options 1 and 2. Rather briefly, so I suggest you now take time out and have a go at creating your own sound strings.

OPTION 3. RAMBLINGS THROUGH ROM/RAM.

Option 1 allowed us to put selected values in memory to store as data for the M/L routine to act on. Those values are stored in strings as part of the BASIC listing. Every table of data used by the M/L routine must have a Ø as its last value to indicate the end of data. When we keyed in the program we purposely left out the end quotation mark from the strings in the utility, and simply pressed ENTER/NEW LINE after keying in an even number of characters. The ENTER/NEW LINE key caused a Ø to be entered as the last byte. All of the 1Ø strings end with a Ø created by this method. If you were able to PEEK at all of ROM (and you can) you will see what appears to be a lot of meaningless numbers, with some of those numbers being Ø. Option 3 allows us to take advantage of the values in ROM and use them as data for our M/L routine. Look at the data in line 18Ø of this listing - there we see a series of numbers including a Ø or two and those numbers are POKEd into high memory (starting at whatever was set for MEMORY SIZE) by line 6Ø. Select Option 3 and we'll see what we can do with those numbers. Wouldn't you know it? More options! We will jump the gun here and select Option 2. You should now see the prompt "START ADDRESS OF ROUTINE ?" on the screen. Type in whatever was set for MEMORY SIZE then press ENTER/NEW LINE. We are then presented with the message "X" = USR (*****)" and more special keys, (unshifted). Press S. The sound isn't the greatest but what we did was to get the M/L routine to read part of itself as data. If it can do that, what about all those numbers and Ø's in ROM? Press 'V' then enter 'Ø' -- X = USR (Ø), then press 'S'. Not the best - but let's keep going. Press 'V' again then enter '4' --- 'X = USR (4)', Not bad! We must be able to use that sound for something. Try entering the following numbers; 7, 66, 73, 1Ø1, 1ØØØ, 11Ø1, 1213, 1383, 4238, 5Ø15, 563Ø ---- but why should I tell you all the good numbers? Press 'R' to return to the options then select 1. We are now going to let the program search through memory for some possible sound routines. Enter Ø for the start address and (say) 1ØØØ for the end address. How MANY NOTES? THis option allows you to enter 1, 2, 3 or any other reasonable value. We'll try 1 for starters. We now see on the screen 'X = USR (4)' with yet more special keys. Press 'S' and that is the sound you will get if you have 'X = USR (4)' in your program. Press 'C'. Now we have 'X = USR(7)'. We've already tried that so continue pressing 'C' & 'S' alternately until you are convinced that you have always had those sound routines in ROM but with no way of using them (until now).

Go back to the options by pressing 'R' and try the routine again, only this time enter 2, 5, 8, 15 or whatever number pleases you in response to 'HOW MANY NOTES?'. You will discover that some of the sounds are quite frightful. And don't be surprised if you don't get any sound at all. The program searches through ROM from the START ADDRESS until a Ø is found, it then deducts 2 for one note, (4 for two notes, 6 for three, etc.) from the memory address of the Ø byte and uses that address for the USR function. For one note we have 1 byte for duration, 1 byte for frequency and Ø end byte. So much for Option 3 and all it's sub-options.

Although this program is designed primarily to help you create your own sound effects for your BASIC programs, it is also a fun program. Play with the program with your friends and see who can create the best (or worst) sound effects. Or try your hand at writing tunes. If you create really good sound effects then MICRO-80 might like to publish them for others to use. You might like to drive your spouse (parents/guardian/children/friends) insane while you try this one:

Select Option 3 - RAMBLINGS THROUGH ROM, then Option 2 - PLAY TUNES FROM MEMORY. Enter the number 14464 then press 'S'. Big deal! You can't hear a thing. While holding down the 'S' key press the SHIFT key. So what? - Now comes the gimmick. Keep the 'S' key and SHIFT Key depressed (with one hand) and use the other hand to press any other key (or combination of keys). Weird isn't it? Try it without the SHIFT key - nothing, not a peep! You may like to do some homework and find out all about memory location 14464.

The 6Ø byte maximum on the string length was only imposed because of the limits of the VDU screen. There are a couple of methods you can use if you want longer strings. One method is presented here: Make both strings 7 & 8 equal to 6Ø bytes then amend the program to read " 16Ø SS$(9) = SS$(7) + SS$(8) + CHR$(Ø) " then whenever you use X = USR (SA(9)) in your program your sound effects will consist of 12Ø notes, and of course, you can still use X = USR(SA(7)) and X = USR(SA(8)) as separate sound effects. After you have created strings 7 & 8 you will have to RUN the program again to re-initialize string 9 (SS$(9)). And, of course, you will have to CLEAR more string space because SS$(9) will be stored in high memory. Those of you with more than 16K of memory will have to adjust the value of SA(9) by subtracting 65536.

Some final words: Don't edit the strings using Level II Edit mode - use Option 1. Don't be surprised by the strange listing you may get after you have created your sounds. There is no need to set MEMORY SIZE, the program does it for you.

Oh! By the way, the program offers a special bonus for the more experienced programmer - use it to assemble your Super Graphics.

Good Luck!

P.S. This program was written before the March issue of MICRO-80 was released. It does not conform with the requests made in the Editorial on page 2, but the program is totally compatible with 16K or more TRS-80 Level II, the System '80 and Disk Basic. The only requirement from the user is to CLOAD & RUN. For those of you who wish to follow the other methods, the M/L routine is <u>completely relocatable</u> without any modifications - put it wherever you like.

```
10 MS=256*PEEK(16562)+PEEK(16561)-60
20 M1=INT(MS/256):M2=MS-256*M1:POKE16562,M1:POKE16561,M2
30 CLS:PRINT@522,"MEMORY SIZE HAS BEEN SET AT";MS
40 CLEAR50:ML=PEEK(16562)*256+PEEK(16561)+2
50 M1=INT(ML/256):M2=ML-M1*256:IFML>32767THENML=ML-65536
60 FORL=MLTOML+47:READB:POKEL,B:NEXT:IFPEEK(16396)=201THENPOKE16
526,M2:POKE16527,M1:ELSECMD"T":DEFUSR0=ML:POKE14308,1
70 IFML<0THENML=ML+65536
80 SS$(0)="RONALD
90 SS$(1)="JAMES  SULLY
100 SS$(2)="------------------
110 SS$(3)=" # $ % & ' ( ) * + , - .
120 SS$(4)=" ? @ A B C D E F G H I J K L M
130 SS$(5)="! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! !
140 SS$(6)=" FD FD FD FD FD FD FD FD FD FD FD FD FD FD
150 SS$(7)=" SDFGHASDFGHASDFGHASDFGHASDFGHASDFGHASDFGHASDFGH
160 SS$(8)="!#$%&'()*=-:@@+;?>.<,ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`a
170 SS$(9)="****************************************************
********
180 FORL=0TO9:K=VARPTR(SS$(L)):SA(L)=PEEK(K+2)*256+PEEK(K+1):NEX
T
190 DATA205,127,10,229,221,225,221,78,0,121,183,200,221,70,1,62,
5,211,255,16,254,221,70,1,62,6,211,255,16,254,13,32,235,221,35,22
1,35,1,255,255,33,48,0,9,56,253,24,214
200 '
WHEN DONE WITH COMPILING YOUR SOUND STRINGS ETC, DELETE THIS
LINE TO THE LAST LINE.  THEN USE A MONITER (E.G. BMON) TO MERGE
YOUR PROGRAM WITH LINES 10-190 OF THIS PROGRAM.
210 '
AMEND YOUR PROGRAM TO INCLUDE X=USR(nnnn)   OR   X=USR(SA(n))
WHEREVER YOU WANT THAT PARTICULAR SOUND.
```

```
220 '
         SOUND EFFECTS REVISITED
         COPYRIGHT (C) APRIL 1981
         RONALD J. SULLY
         1 PACKHAM PLACE
         CANBERRA ACT 2615
         (062) 582917

230 IFML<32767THENTM=32767ELSEIFML<49151THENTM=49151ELSETM=65535
240 DIMBV(60),PA(60):P=256:FORL=1TO60STEP20:FORM=LTOL+19STEP2:FO
RN=MTOM+1:PA(N)=P:P=P+10:NEXTN:P=P+44
250 NEXTM:P=P-618:NEXTL:U1$="##":U2$="###":AR$=CHR$(93)+"<"
260 X=USR(2000):CLS
270 PRINT"
                    SOUND EFFECTS REVISITED
                    ======================
                     (OR $TRING$ & THING$)
280 X=USR(4):PRINT"
                            OPTIONS:
"
290 PRINT"
                    1.  BUILD OR EDIT $TRING
                    2.  PLAY $TRING NOTES
                    3.  RAMBLINGS THROUGH ROM/RAM
                    4.  END
"
300 PRINT"                SELECT  1, 2, 3 OR 4:"
310 X=USR(4):PRINT@867,CHR$(140);:S=VAL(INKEY$)
320 IFS<1ORS>4THENFORT=1TO100:NEXT:PRINT@867," ";:FORT=1TO100:NE
XT:GOTO310ELSECLS:ONSGOTO330,620,690,950
330 X=USR(270):PRINT" BUILD $TRING OPTION:
<VALUES TO BE KEPT IN RANGE 1-255> - VALUE 34 INVALID"
340 PRINT"WHICH $TRING:  <0 - 9>":I$=INKEY$
350 X=USR(66):I$=INKEY$:IFI$=""THEN350ELSES=VAL(I$):PRINT@152,S:
LE=LEN(SS$(S))
360 IFLE>60CLS:PRINT"ERROR ---  SS$(";S;") IS  GREATER   THAN  6
0 CHARACTERS; IT IS";LE;"CHARACTERS LONG: FIX IT!":LIST70-160
370 I=0:INPUT"INCREMENT/DECREMENT <VALUE OR ENTER>";I:X=USR(492)
:CLS
380 PRINT@0," <a>BORT:    <c>OPY:    <d>ECREMENT:    <e>XIT:
<i>NCREMENT:  <p>ROPORTIONAL:    <r>ANDOM:    <s>AME:"
390 PRINT@132,"$TRING NUMBER";S;TAB(45)"INC/DEC =";I
400 PRINTTAB(2);"DUR";TAB(12);"FREQ";TAB(24);"DUR";TAB(34);"FREQ
";TAB(46);"DUR";TAB(56);"FREQ"
410 FORL=213TO853STEP64:PRINT@L,CHR$(149);:PRINT@L+22,CHR$(149);
:NEXT:FORL=1TOLE:X=USR(66)
420 PRINT@PA(L),USINGU1$;L;:PRINT":";:NEXT
430 FORL=0TOLE-1:BV(L+1)=PEEK(SA(S)+L):PRINT@PA(L+1)+3,USINGU2$;
BV(L+1);:X=USR(110):NEXT:I$=INKEY$:M=1
440 B$="":L=0
450 IFBV(L+1)>0ANDBV(L+1)<256ANDBV(L+1)<>34THENPRINT@960,CHR$(20
2);"S O U N D  E F F E C T S  R E V I S I T E D";CHR$(202);
460 I$=INKEY$:X=USR(888):PRINT@PA(M)+7,"  ";:IFPEEK(14400)=8ANDB
$=""THENM=M-1:GOTO570
470 IFPEEK(14400)=16ANDB$=""THENM=M+1:GOTO570
480 IFI$>CHR$(47)ANDI$<CHR$(58)THENB$=B$+I$:I$="":PRINT@PA(M)+3,
"   ";:PRINT@PA(M)+3,B$;:BV(M)=VAL(B$):IFLEN(B$)=3THENB$="":M=M+1
:GOTO570
490 IFI$>CHR$(31)ANDI$<CHR$(91)THENBV(M)=ASC(I$):X=USR(2102):GOT
O600ELSEIFI$="a"THENX=USR(1381):GOTO430
500 IFI$="c"ANDM>2THENX=USR(66):BV(M)=BV(M-2):GOTO600
510 IFI$="d"ANDM>2THENX=USR(1215):BV(M)=BV(M-2)-I:GOTO600
520 IFI$="e"THENX=USR(4):FORL=0TOLE-1:IFBV(L+1)<1ORBV(L+1)>255OR
BV(L+1)=34THEN680ELSEPOKESA(S)+L,BV(L+1):NEXT:GOTO260
530 IFI$="i"ANDM>2THENX=USR(1000):BV(M)=BV(M-2)+I:GOTO600
540 IFI$="p"ANDM<>1THENX=USR(270):BV(M)=256-BV(M-1):GOTO600
550 IFI$="r"THENX=USR(955):BV(M)=RND(255):GOTO600ELSEIFI$="s"AND
M<>1THENX=USR(492):BV(M)=BV(M-1):GOTO600
560 IFI$=CHR$(13)ANDB$=""THENX=USR(66):M=M+1:GOTO570ELSEIFI$=CHR
$(13)ANDB$<>""THENBV(M)=VAL(B$):GOTO610
570 IFM<1THENM=LE
580 IFM>LETHENM=1
590 PRINT@PA(M)+7,AR$;:GOTO450
600 B$=RIGHT$(STR$(BV(M)),3)
```

```
610 IFLEN(B$)<3THENB$=" "+B$:GOTO610ELSEPRINT@PA(M)+3,B$;:M=M+1:
B$="":GOTO570
620 X=USR(1783)
630 PRINT"
THIS ROUTINE ALLOWS YOU TO PLAY ANY ONE OF THE SOUND ROUTINES
BUILT UP VIA OPTION 1.

TO HEAR THE NOTES PRESS THE RELEVANT NUMBER KEY  <0 - 9>"
640 PRINT@768,"<R> TO RETURN TO THE OPTIONS":I$=INKEY$
650 PRINT@591,"USE  ";CHR$(94);" X=USR(SA( )) ";CHR$(93);" IN YO
UR PROGRAM";
660 I$=INKEY$:IFI$=""THEN660ELSEIFI$="R"THEN260ELSES=VAL(I$)
670 IFPEEK(15359)=0THEN660ELSEPRINT@607,RIGHT$(STR$(S),1);:X=USR
(SA(S)):GOTO670
680 PRINT@960,"VALUE IN BYTE";L+1;"IS OUTSIDE LIMIT <1-255>  OR
= 34 :-- REDO";:M=L+1:GOTO450
690 X=USR(1253):CLS
700 PRINTTAB(15)"SOUND EFFECTS REVISITED

          OPTIONS:

          1.  SEARCH MEM FOR SOUND ROUTINES
          2.  PLAY TUNES FROM MEMORY (THE 80'S NOT YOURS!)
          3.  RETURN TO MAIN OPTIONS

          SELECT 1, 2  OR 3:"
710 PRINT@539,CHR$(140);:I$=INKEY$
720 IFI$=""THENX=USR(4):FORT=1TO100:NEXT:PRINT@539," ";:FORT=1TO
100:NEXT:GOTO710
730 I=VAL(I$):IFI<1ORI>3THEN710ELSEONIGOTO740,860,260
740 CLS
750 PRINT"SEARCH MEMORY ROUTINE:

YOU WILL NEED TO INPUT THREE NUMBERS;
          START ADDRESS   <NOT LESS THAN 0>
          END ADDRESS     <NOT GREATER THAN TOP OF MEMORY>
          NUMBER OF NOTES <KEEP IT REASONABLE>"
760 INPUT"START ADDRESS";S:X=USR(492):IFS<0THEN730ELSEINPUT"END
ADDRESS";E:X=USR(492):IFE>TMTHEN740
770 INPUT"HOW MANY NOTES";N:X=USR(492):N=N*2:FORL=STOE:K=L:IFK>3
2767THENK=K-65536
780 B=PEEK(K):M=K-N:IFM<0ANDL<32768THEN840ELSEM$=STR$(M)
790 PRINT@653,"USE ";CHR$(94);"  X=USR(";M$;") ";CHR$(93);" IN Y
OUR PROGRAM.";
800 IFB=0THENI$=INKEY$:PRINT@704,"PRESS
<S> TO SOUND THIS NOTE
<C> TO CONTINUE THE SEARCH
<R> TO RETURN TO OPTIONS";ELSE840
810 I$=INKEY$:IFI$=""THEN810ELSEIFI$="C"THENPRINT@704,CHR$(31);:
GOTO840ELSEIFI$="R"THEN690
820 IFI$<>"S"THEN810
830 IFPEEK(15359)=0THEN810ELSEX=USR(M):GOTO830
840 NEXT:PRINT@640,CHR$(31);:PRINT@960,"END OF SEARCH  - PRESS A
NY KEY TO RETURN TO OPTIONS";
850 IFINKEY$=""THEN850ELSE690
860 CLS
870 PRINT"
THIS ROUTINE ALLOWS YOU TRY SPECIFIC SOUND ROUTINES THAT YOU
FOUND IN MEMORY FROM OPTION 1.
OR SIMPLY ENTER ANY MEMORY LOCATION FROM 0 TO TOP OF MEMORY
AND TRY YOUR LUCK.
"
880 PRINT@448,CHR$(31);:PRINT@448,"";:INPUT"START ADDRESS OF ROU
TINE";E:IFE>TMTHEN880
890 IFE>32767THENE=E-65536
900 S$=STR$(E)+")":PRINT@448,CHR$(31)
910 PRINT@590,"USE ";CHR$(94);" X=USR(";S$;" ";CHR$(93);" IN YOU
R PROGRAM";
920 PRINT@704,"PRESS
<S> TO HEAR THE SOUND
<V> TO INPUT ANOTHER START ADDRESS
<R> TO RETURN TO OPTIONS"
930 P=PEEK(14340):IFP=8THENPRINT@704,CHR$(31);:X=USR(E):GOTO920E
LSEIFP=4THEN690
940 IFP=64THENPRINT@576,CHR$(31):GOTO880ELSE930
950 X=USR(1361):FORL=0TO9:PRINT@470,"X=USR(SA(";L;"))";:X=USR(SA
(L)):NEXT:CLS:END
```

***** NEXT MONTH'S ISSUE *****

Next month's issue will contain at least the following programs plus the usual features and articles.

** POKER LII/16K **

Now you can play poker on your '80. This game draws life size cards on the screen, lets you bet on your hand and, if you win it, gives you the option to double up, i.e. on the turn of a card if you get a card greater than eight, you win twice your original winnings.

** THE TOWERS OF HANOI LII/16K **

This is based on the age-old game of the same name, which has three pegs and a number of different sized discs, (not floppy ones either). The player has to get all the discs from one peg to another without placing a large disc on top of a smaller one.

** PUT m/l 4K **

In response to readers' requests to be able to SET non graphics, this program does just that. No muss, no fuss, simply type....PUT(X,Y),"stringname" and its done.

** SHIFTLOCK m/l 4K **

A program for the year of the disabled. This is a program that was created to allow a disabled person working with a stylus (or one finger) to enter shifted key values.

** MORSE CODE DECODER LII/4K **

This month you can learn semaphore signals and next month you can learn what all those dots and dashes stand for. With this morse coder decoder, you just type in dots and dashes and the computer comes back with a letter.

** SHORTEN 32K/DISK **

This program will go through your basic program and remove all unnecessary blanks, REMark statements and linefeeds. A real life saver for those times when every byte counts.

APPLICATION FOR PUBLICATION OF A PROGRAM IN MICRO-80

Date ....

Tick where appropriate

To **MICRO-80**
Please consider the enclosed program for . . .

(i) Publication in MICRO-80 ....

(ii) Publication on disk or cassette only ....

(iii) Both ....

Name ....

Address ....

....

....Postcode ....

* * * CHECK LIST * * *

Please ensure that the cassette or disk is clearly marked with your name and address, program name(s), Memory size, Level I, II, System 1 or 2, Edtasm, System, etc. The use of REM statements with your name and address is suggested, in case the program becomes separated from the accompanying literature.

Ensure that you supply adequate instructions, notes on what the program does and how it does it, etc.

For system tapes, the start, end, and entry points, etc.

The changes or improvements that you think may improve it.

Please package securely — padabags are suggested — and enclose stamps or postage if you want your cassette or disk returned.

***** CASSETTE EDITION INDEX *****

The cassette edition of MICRO-80 contains all the software listed each month, on cassette. All cassette subscribers need do is CLOAD and RUN the programs. Level II programs are recorded on side 1 of the cassette. Level I programs are recorded on side 2. Level I programs are not compatible with the System 80. All programs are recorded twice in succession. Note, System 80 computers have had different tape-counters fitted at different times. The approximate start positions shown are correct for the very early System 80 without the volume control or level meter. They are probably incorrect for later machines. The rates for a cassette subscription are printed on the inside front cover of each issue of the magazine.

The disk edition contains all those programs which can be executed from disk, including Level I programs. Level I disk programs are saved in the NEWDOS format. Users require the Level I/CMD utility supplied with NEWDOS + or NEWDOS 80 version 1.0 to run them.

				APPROX. START POSITION		
SIDE ONE	TYPE	I.D.	DISK FILESPEC	CTR-41	CTR-80	SYSTEM-80
SEMAPHORE	LII/4K	S	SEMAPHOR/BAS	7	5	5
"	"	"	"	37	25	26
THREE BILLY GOATS GRUFF	LII/16K	G	GOATS/BAS	66	45	47
"	"	"	"	109	74	78
SOLITAIRE	LII/16K	A	SOLITAIR/BAS	150	101	106
"	"	"	"	175	118	124
MOVIE	SYSTEM	MOVIE	MOVIE/CMD	200	135	142
"	"	"		205	139	146
"	EDTASM	"	MOVIE/EDT	211	143	150
"	"	"		233	158	166
BASIC ARRAY SAVER	LII/4K	B	ARRAYSAV/BAS	254	172	180
"	"	"		294	199	210
ESF LOWER CASE DRIVER	LII/4K	E	ESFLCD/BAS	332	225	237
"	"	"		340	230	242
SIDE TWO						
ROVING TARGETS	LI/4K	-	ROVING/LV1	15	10	-
"	"	-		77	52	-
SOUND EFFECTS REVISITED	LII/DISK/16K	C	SER/BAS	134	91	96
"	"	"		183	124	130
DEMONSTRATION OF "ORCHESTRA 80" MUSIC		AUDIO		240	160	168

TO:
MICRO-80, P.O. BOX 213, GOODWOOD, SOUTH AUSTRALIA. 5034.

Please RUSH to me the items shown below:

$ .... enclosed Date ....

....................12 month subscription to MICRO-80

....................12 month subscription to MICRO-80, plus the cassette edition

....................The latest issue of MICRO-80 (see inside front cover for prices)

The MICRO-80 PRODUCTS listed below:

DESCRIPTION	PRICE

TOTAL ENCLOSED WITH ORDER

☐ Cheque ☐ Bankcard ☐ Money Order

Bankcard Account Number

Signature .... Exp. End ....

NAME ....

ADDRESS ....

.... Postcode ....

bankcard

A SPECIAL OFFER!

TO

SUBSCRIBERS OF **MICRO 80**

FOR THE FIRST TIME IN AUSTRALIA!
GENUINE TANDY TRS–80 MICRO-COMPUTERS, HARDWARE AND SOFTWARE AT

GREAT DISCOUNT PRICES!!!

HOW TO ORDER –

1. SELECT ITEMS FROM 1980 TANDY CATALOGUE
2. DEDUCT 10% FROM ADVERTISED PRICES
3. POST US YOUR ORDER STATING DESCRIPTION CAT. No AND A CHEQUE OR MONEY ORDER.

WE WILL –

1. ATTEND TO YOUR ORDER WITHIN 7 DAYS *
2. SUPPLY GOODS SELECTED FREIGHT FREE !
3. SEND ADVERTISING REGULARLY TO KEEP YOU INFORMED OF CURRENT SPECIALS !

* *subject to availability*

to **CONQUEST ELECTRONICS** Pty. Ltd.
212 Katoomba St. KATOOMBA 2780

Please supply –

QTY.	CAT NO	DESC.	ADV. PRICE
		SUB TOTAL	
		LESS 10%	
		FIND CHEQUE FOR **TOTAL**	

SEND **FREIGHT FREE** TO

NAME ....
ADDRESS ....
.... P/ Code ....

TANDY ELECTRONICS DEALER

CONQUEST ELECTRONICS PTY. LTD.

212 KATOOMBA ST KATOOMBA N.S.W. 2780 PHONE (047) 82 2491

MICRO-80

LEVEL II ROM REFERENCE MANUAL

by Edwin Paay

Published by MICRO-80 PRODUCTS

Written by Eddy Paay, the LEVEL II ROM REFERENCE MANUAL is the most complete explanation of the Level II BASIC interpreter ever published.

Part 1 lists all the useful and usable ROM routines, describes their functions explains how to use them in your own machine language programs and notes the effect of each on the various Z 80 registers.

Part 1 also details the contents of system RAM and shows you how to intercept BASIC routines as they pass through system RAM. With this knowledge, you can add your own commands to BASIC, for instance, or position BASIC programs in high memory—the only restriction is your own imagination!

Part 2 gives detailed explanations of the processes used for arithmetical calculations, logical operations, data movements, etc. It also describes the various formats used for BASIC, SYSTEM and EDITOR/ASSEMBLER tapes. Each section is illustrated by sample programs which show you how you can use the ROM routines to speed up your machine language programs and reduce the amount of code you need to write.

The LEVEL II ROM REFERENCE MANUAL is intended to be used by machine language programmers. It assumes a basic understanding of the Z 80 instruction set and some experience of. Assembly Language programming. But BASIC programmers too will benefit from reading it. They will gain a much better insight into the functioning of the interpreter which should help them to write faster, more concise BASIC programs.

MICRO-80